```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III      VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III      VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDD          DDD RRR       RRR          III        VVV     VVV    EEE              RRR       RRR
DDDDDDDDDDDD     RRR           RRR  IIIIIIIII         VVV       EEEEEEEEEEEEEEE  RRR           RRR
DDDDDDDDDDDD     RRR           RRR  IIIIIIIII         VVV       EEEEEEEEEEEEEEE  RRR           RRR
DDDDDDDDDDDD     RRR           RRR  IIIIIIIII         VVV       EEEEEEEEEEEEEEE  RRR           RRR
```

```
DDDDDDDD    UU      UU  HH      HH    IIIIII    RRRRRRRR    TTTTTTTTTT
DDDDDDDD    UU      UU  HH      HH    IIIIII    RRRRRRRR    TTTTTTTTTT
DD      DD  UU      UU  HH      HH      II      RR      RR      TT
DD      DD  UU      UU  HH      HH      II      RR      RR      TT
DD      DD  UU      UU  HH      HH      II      RR      RR      TT
DD      DD  UU      UU  HH      HH      II      RR      RR      TT
DD      DD  UU      UU  HHHHHHHHHH      II      RRRRRRRR        TT
DD      DD  UU      UU  HHHHHHHHHH      II      RRRRRRRR        TT
DD      DD  UU      UU  HH      HH      II      RR   RR         TT
DD      DD  UU      UU  HH      HH      II      RR   RR         TT
DD      DD  UU      UU  HH      HH      II      RR    RR        TT          ....
DD      DD  UU      UU  HH      HH      II      RR    RR        TT          ....
DDDDDDDD    UUUUUUUUUU  HH      HH    IIIIII    RR      RR      TT          ....
DDDDDDDD    UUUUUUUUUU  HH      HH    IIIIII    RR      RR      TT          ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  DUHIRT HOST INITIATED REPLACEMENT FOR THE DISK CLASS DRIVER
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27
0000    28  ;++
0000    29  ;
0000    30  ; FACILITY:
0000    31  ;
0000    32  ;       MSCP Disk Class Driver
0000    33  ;
0000    34  ; ABSTRACT:
0000    35  ;
0000    36  ;       Buddy!  You're in a world of HIRT (Host Inititated Replacement Table).
0000    37  ;
0000    38  ;       This module contains all the routines and data structure definitions
0000    39  ;       needed by the disk class driver to perform host initiated replacement
0000    40  ;       of questionable blocks on disks conforming to the DSA specification.
0000    41  ;
0000    42  ; ENVIRONMENT:
0000    43  ;
0000    44  ;       This module is linked into DUDRIVER, the VMS disk class driver.
0000    45  ;
0000    46  ;--
0000    47  ;
0000    48  ; AUTHOR: Ralph O. Weber (ghost writer for Robert L. Rappaport)
0000    49  ;
0000    50  ; CREATION DATE: 21-JAN-1984
0000    51  ;
0000    52  ; MODIFIED BY:
0000    53  ;
0000    54  ;       V03-004 ROW0398         Ralph O. Weber          21-JUL-1984
0000    55  ;               Setup use of class driver write-lock bit in UCB$W_DEVSTS.
0000    56  ;               Also eliminate alteration and use of DEV$V_SWL bit in
0000    57  ;               UCB$L_DEVCHAR.  That bit is controlled by the file system.
```

```
0000     58 ;
0000     59 ;       V03-003 ROW0346         Ralph O. Weber          11-APR-1984
0000     60 ;               > Add worst failure status reporting to insure that I/O
0000     61 ;                 requests producing failed replacement requests get failure
0000     62 ;                 status codes.
0000     63 ;               > Add several more error logging points.
0000     64 ;               > Supress error recovery and error correction when testing
0000     65 ;                 the possibly bad block in step 7.
0000     66 ;               > Use cheep class driver macros for common functions like
0000     67 ;                 testing MSCP success/failure and initializing a MSCP command
0000     68 ;                 packet.
0000     69 ;
0000     70 ;       V03-002 ROW0332         Ralph O. Weber           2-APR-1984
0000     71 ;               Cause message to appear on the system console whenever an
0000     72 ;               error occurs during RDT processing.
0000     73 ;
0000     74 ;       V03-001 ROW0331         Ralph O. Weber          29-MAR-1984
0000     75 ;               Change DU$CANCEL_FROM_HIRT to count wait count adjustment in
0000     76 ;               CDRP$W_DUTUCNTR.  Also add comments from old DUDRIVER to its
0000     77 ;               module header.
0000     78 ;
0000     79 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; INCLUDE FILES:
                ;
                        $CDDBDEF                        ;Define CDDB offsets
                        $CDRPDEF                        ;Define CDRP offsets
                        $CDTDEF                         ;Define CDT offsets
                        $CRBDEF                         ;Define CRB offsets
                        $DDBDEF                         ;Define DDB offsets
                        $DEVDEF                         ;Define DEVICE CHARACTERISTICS bits
                        $DYNDEF                         ;Define DYN symbols
                        $EMBLTDEF                       ;Define EMB Log Message Types
                        $FKBDEF                         ;Define FKB offsets
                        $IODEF                          ;Define I/O FUNCTION codes
                        $IPLDEF                         ;Define IPL levels
                        $IRPDEF                         ;Define IRP offsets
                        $MSCPDEF                        ;Define MSCP packet offsets
                        $MSLGDEF                        ;Define MSCP Error Log offsets
                        $PBDEF                          ;Define Path Block offsets
                        $PCBDEF                         ;Define PCB offsets
                        $PDTDEF                         ;Define PDT offsets
                        $PRDEF                          ;Define Processor Registers
                        $RCTDEF                         ;Define RCT offsets
                        $RDDEF                          ;Define RDTE offsets
                        $SBDEF                          ;Define System Block Offsets
                        $SCSCMGDEF                      ;Define SCS Connect Message offsets
                        $SSDEF                          ;Define System Status values
                        $UCBDEF                         ;Define UCB offsets
                        $VADEF                          ;Define Virtual Address offsets
                        $VECDEF                         ;Define INTERRUPT DISPATCH VECTOR offsets

                        $DUTUDEF                        ;Define common class driver CDDB
                                                        ; extensions and other common symbols


                ; CONSTANTS

B6DBCB6D        TEST_PATTERN=^xB6DBCB6D                 ; Pattern to write on bad blocks.

                ;
                ; MODULE PSECT
                ;
                        .PSECT  $$$115_DRIVER LONG
                ;
                ; SET DEFAULT DISPLACEMENT
                ;
                        .DEFAULT DISPLACEMENT WORD
```

```
0000   130          .SBTTL  MACRO DEFINITIONS
0000   131
0000   132          .MACRO  HIRT_SUBSAVE                    ; Save return on HIRT substack.
0000   133          POPL    @HIRT$L_STKPTR                  ; Pop return from stack onto substack.
0000   134          ADDL    #4,HIRT$L_STKPTR                ; Bump substack pointer.
0000   135          .ENDM   HIRT_SUBSAVE
0000   136
0000   137          .MACRO  HIRT_SUBUNSAVE                  ; Pop top of SUBSTACK and push onto stack.
0000   138          SUBL    #4,HIRT$L_STKPTR                ; Pop substack.
0000   139          PUSHL   @HIRT$L_STKPTR                  ; Put top of substack onto top of stack.
0000   140          .ENDM   HIRT_SUBUNSAVE
0000   141
0000   142          .MACRO  HIRT_SUBRETURN                  ; HIRT_SUBUNSAVE and return to caller.
0000   143          HIRT_SUBUNSAVE
0000   144          RSB                                     ; Return to subroutine caller.
0000   145          .ENDM   HIRT_SUBRETURN
0000   146
0000   147 ;
0000   148 ; Expanded opcode macros - Branch word conditional psuedo opcodes.
0000   149 ;
0000   150
0000   151 ;
0000   152 ; BWNEQ - Branch (word offset) not equal
0000   153 ;
0000   154
0000   155          .MACRO  BWNEQ   DEST,?L1
0000   156          .SHOW
0000   157          BEQL    L1                      ; Branch around if NOT NEQ.
0000   158          BRW     DEST                    ; Branch to destination if NEQ.
0000   159 L1:                                     ; Around.
0000   160          .NOSHOW
0000   161          .ENDM   BWNEQ
0000   162
0000   163 ;
0000   164 ; BWEQL - Branch (word offset) equal
0000   165 ;
0000   166
0000   167          .MACRO  BWEQL   DEST,?L1
0000   168          .SHOW
0000   169          BNEQ    L1                      ; Branch around if NOT EQL.
0000   170          BRW     DEST                    ; Branch to destination if EQL.
0000   171 L1:                                     ; Around.
0000   172          .NOSHOW
0000   173          .ENDM   BWEQL
0000   174
0000   175 ;
0000   176 ; BWBS - Branch (word offset) bit set.
0000   177 ;
0000   178
0000   179          .MACRO  BWBS    BIT,FIELD,DEST,?L1
0000   180          .SHOW
0000   181          BBC     BIT,FIELD,L1                    ; Branch around if bit NOT set.
0000   182          BRW     DEST                            ; Branch to destination if bit set.
0000   183 L1:                                             ; Around.
0000   184          .NOSHOW
0000   185          .ENDM   BWBS
0000   186
```

```
0000   187 ;
0000   188 ; BWBC - Branch (word offset) bit clear.
0000   189 ;
0000   190
0000   191          .MACRO  BWBC    BIT,FIELD,DEST,?L1
0000   192          .SHOW
0000   193          BBS     BIT,FIELD,L1            ; Branch around if bit NOT clear.
0000   194          BRW     DEST                    ; Branch to destination if bit clear.
0000   195 L1:                                     ; Around.
0000   196          .NOSHOW
0000   197          .ENDM   BWBC
```

```
                        .SBTTL  IRP - CDRP Consistancy Check

                ; The following set of ASSUME statements will all be true as long as
                ;       the IRP and CDRP definitions remain consistent.

                        ASSUME  CDRP$L_IOQFL-CDRP$L_IOQFL       EQ      IRP$L_IOQFL
                        ASSUME  CDRP$L_IOQBL-CDRP$L_IOQFL       EQ      IRP$L_IOQBL
                        ASSUME  CDRP$W_IRP_SIZE-CDRP$L_IOQFL    EQ      IRP$W_SIZE
                        ASSUME  CDRP$B_IRP_TYPE-CDRP$L_IOQFL    EQ      IRP$B_TYPE
                        ASSUME  CDRP$B_RMOD-CDRP$L_IOQFL        EQ      IRP$B_RMOD
                        ASSUME  CDRP$L_PID-CDRP$L_IOQFL         EQ      IRP$L_PID
                        ASSUME  CDRP$L_AST-CDRP$L_IOQFL         EQ      IRP$L_AST
                        ASSUME  CDRP$L_ASTPRM-CDRP$L_IOQFL      EQ      IRP$L_ASTPRM
                        ASSUME  CDRP$L_WIND-CDRP$L_IOQFL        EQ      IRP$L_WIND
                        ASSUME  CDRP$L_UCB-CDRP$L_IOQFL         EQ      IRP$L_UCB
                        ASSUME  CDRP$W_FUNC-CDRP$L_IOQFL        EQ      IRP$W_FUNC
                        ASSUME  CDRP$B_EFN-CDRP$L_IOQFL         EQ      IRP$B_EFN
                        ASSUME  CDRP$B_PRI-CDRP$L_IOQFL         EQ      IRP$B_PRI
                        ASSUME  CDRP$L_IOSB-CDRP$L_IOQFL        EQ      IRP$L_IOSB
                        ASSUME  CDRP$W_CHAN-CDRP$L_IOQFL        EQ      IRP$W_CHAN
                        ASSUME  CDRP$W_STS-CDRP$L_IOQFL         EQ      IRP$W_STS
                        ASSUME  CDRP$L_SVAPTE-CDRP$L_IOQFL      EQ      IRP$L_SVAPTE
                        ASSUME  CDRP$W_BOFF-CDRP$L_IOQFL        EQ      IRP$W_BOFF
                        ASSUME  CDRP$L_BCNT-CDRP$L_IOQFL        EQ      IRP$L_BCNT
                        ASSUME  CDRP$W_BCNT-CDRP$L_IOQFL        EQ      IRP$W_BCNT
                        ASSUME  CDRP$L_IOST1-CDRP$L_IOQFL       EQ      IRP$L_IOST1
                        ASSUME  CDRP$L_MEDIA-CDRP$L_IOQFL       EQ      IRP$L_MEDIA
                        ASSUME  CDRP$L_IOST2-CDRP$L_IOQFL       EQ      IRP$L_IOST2
                        ASSUME  CDRP$L_TT_TERM-CDRP$L_IOQFL     EQ      IRP$L_TT_TERM
                        ASSUME  CDRP$B_CARCON-CDRP$L_IOQFL      EQ      IRP$B_CARCON
                        ASSUME  CDRP$Q_NT_PRVMSK-CDRP$L_IOQFL   EQ      IRP$Q_NT_PRVMSK
                        ASSUME  CDRP$L_ABCNT-CDRP$L_IOQFL       EQ      IRP$L_ABCNT
                        ASSUME  CDRP$W_ABCNT-CDRP$L_IOQFL       EQ      IRP$W_ABCNT
                        ASSUME  CDRP$L_OBCNT-CDRP$L_IOQFL       EQ      IRP$L_OBCNT
                        ASSUME  CDRP$W_OBCNT-CDRP$L_IOQFL       EQ      IRP$W_OBCNT
                        ASSUME  CDRP$L_SEGVBN-CDRP$L_IOQFL      EQ      IRP$L_SEGVBN
                        ASSUME  CDRP$L_JNL_SEQNO-CDRP$L_IOQFL   EQ      IRP$L_JNL_SEQNO
                        ASSUME  CDRP$L_DIAGBUF-CDRP$L_IOQFL     EQ      IRP$L_DIAGBUF
                        ASSUME  CDRP$L_SEQNUM-CDRP$L_IOQFL      EQ      IRP$L_SEQNUM
                        ASSUME  CDRP$L_EXTEND-CDRP$L_IOQFL      EQ      IRP$L_EXTEND
                        ASSUME  CDRP$L_ARB-CDRP$L_IOQFL         EQ      IRP$L_ARB
```

```
                        241          .SBTTL  Static Storage
                        242          .SBTTL  - HIRT - Host Initiated Replacement Table
                        243
                        244 ;+
                        245 ; The following table is allocated within the Disk Class Driver.  There is
                        246 ; only one such table per system.  The HIRT is used to control resources
                        247 ; needed by the Host Initiated Replacement of disk blocks algorithms.  In
                        248 ; order to limit the resources dedicated to this activity, only one such
                        249 ; replacement is allowed to proceed at any given instant of time.  Replacement
                        250 ; requests which cannot be immediately satisfied are queued.
                        251 ;-
                        252
                        253                  .SAVE
                        254                  .PSECT  $$$300_HIRT LONG
                        255
00000000  0000        256 HIRT$L_RPLQFL:   .LONG   0               ; Request Queue FLINK.
00000000  0004        257 HIRT$L_RPLQTP:   .LONG   0               ; Request Queue Tail Pointer.
     0000 0008        258 HIRT$W_IOST:     .WORD   0               ; Static storage for routines.
     0000 000A        259 HIRT$W_IOWORST:  .WORD   0               ; Worst I/O status encountered.
     0000 000C        260                  .WORD   0
     0000 000E        261 HIRT$W_STS:      .WORD   0               ; HIRT status word.
                      262
                      263          $VIELD  HIRT,0,<-
                      264                       <ACTIVE,,M>,-      ; Set means HIRT has been initialized.
                      265                       <BUSY,,M>,-        ; Set means HIRT being used currently.
                      266                       <FE,,M>,-          ; Set means force error on original data
                      267                       <MATCH,,M>,-       ; SEARCH RCT bit - set => LBN matched
                      268                       <EMPTYPE,,M>,-     ; SEARCH RCT BIT - set => not primary
                      269                       <RESCAN,,M>,-      ; SEARCH RCT BIT - set => reached NULLS
                      270                       <RCTFULL,,M>,-     ; SEARCH RCT BIT - set => no more RBNs avail
                      271                       <ERLOGIP,,M>,-     ; Error Log message has been generated
                      272                       <RCTFE,,M>,-       ; Write RCT block with Forced Error
                      273                      >
                      274
00000000  0010        275 HIRT$L_LOOPCNT:  .LONG   0               ; Loop count used in READ_RCT_BLOCK and
                      276                                         ;  WRITE_RCT_BLOCK.
00000000  0014        277 HIRT$L_OWNUCB:   .LONG   0               ; If HIRT busy, owner UCB address.
00000000  0018        278 HIRT$L_LBN:      .LONG   0               ; LBN being replaced for UCB.
00000000  001C        279 HIRT$L_SAVDCDRP: .LONG   0               ; CDRP address of I/O request of owner.
00000000  0020        280 HIRT$L_CDRP:     .LONG   0               ; Address of permanent CDRP for replacement.
                      281
00000000  0024        282 HIRT$L_PAGE0PTR: .LONG   0               ; System Virtual Address of scratch page
                      283                                         ;  needed by Replacement algorithm.
00000000  0028        284 HIRT$L_PAGE1PTR: .LONG   0               ; System Virtual Address of scratch page
                      285                                         ;  needed by Replacement algorithm.
00000000  002C        286 HIRT$L_PAGE2PTR: .LONG   0               ; System Virtual Address of scratch page
                      287                                         ;  needed by Replacement algorithm.
00000000  0030        288 HIRT$L_PAGE3PTR: .LONG   0               ; System Virtual Address of scratch page
                      289                                         ;  needed by Replacement algorithm.
                      290
00000000  0034        291 HIRT$L_SVAPTE0:  .LONG   0               ; SVAPTE of page 0.
00000000  0038        292 HIRT$L_SVAPTE1:  .LONG   0               ; SVAPTE of page 1.
00000000  003C        293 HIRT$L_SVAPTE2:  .LONG   0               ; SVAPTE of page 2.
00000000  0040        294 HIRT$L_SVAPTE3:  .LONG   0               ; SVAPTE of page 3.
                      295
     0000 0044        296 HIRT$W_BOFF0:    .WORD   0               ; BOFF of page 0.
     0000 0046        297 HIRT$W_BOFF1:    .WORD   0               ; BOFF of page 1.
```

```
                     0000  0048   298 HIRT$W_BOFF2:   .WORD   0               ; BOFF of page 2.
                     0000  004A   299 HIRT$W_BOFF3:   .WORD   0               ; BOFF of page 3.
                           004C   300
                           004C   301 ; Array of words that give the relative RCT sector number contained in a page.
                           004C   302
                     0000  004C   303 HIRT$W_PG0CNTNT:.WORD   0               ; Page 0 contents.
                     0000  004E   304 HIRT$W_PG1CNTNT:.WORD   0               ; Page 1 contents.
                     0000  0050   305 HIRT$W_PG2CNTNT:.WORD   0               ; Page 2 contents.
                     0000  0052   306 HIRT$W_PG3CNTNT:.WORD   0               ; Page 3 contents.
                           0054   307
                           0054   308 ; Static storage needed by several routines that read and write RCT blocks.
                           0054   309
                     0000  0054   310 HIRT$W_SECTORNO:.WORD   0               ; Sector number.
                     0000  0056   311 HIRT$W_PAGENO:  .WORD   0               ; Page number.
                           0058   312
                           0058   313 ; Static storage needed by SEARCH_RCT subroutine.
                           0058   314
                00000000   0058   315 HIRT$L_RBN:     .LONG   0               ; RBN returned to caller.
                00000000   005C   316 HIRT$L_MATCHRBN:.LONG   0               ; Previous RBN that failed.
                00000000   0060   317 HIRT$L_BADRBND: .LONG   0               ; Bad RBN descriptor contents,
                           0064   318                                         ;  used in STEP15 error recovery.
                00000000   0064   319 HIRT$L_STARTBLK:.LONG   0               ; Sector number of Primary RBN.
                00000000   0068   320 HIRT$L_RCTBLOCK:.LONG   0               ; Current RCT sector number.
                00000000   006C   321 HIRT$L_OFFSET:  .LONG   0               ; Offset into current RCT sector.
                           0070   322
                           0070   323 ; HIRT SUBSTACK - used by single threaded replacement algorithm as a return
                           0070   324 ;       point stack.
                           0070   325
                00000000   0070   326 HIRT$L_STKPTR:  .LONG   0               ; Pointer to top of SUBSTACK.
00000000 00000000 00000000 00000000   0074   327 HIRT$L_SUBSTACK:.LONG   0,0,0,0,0       ; SUBSTACK itself.
                00000000   0084
                00000005   0088   328 HIRT$K_SUBSTKLN=<.-HIRT$L_SUBSTACK>/4   ; Total length of SUBSTACK in longwords.
                           0088   329
                      00000000    330         .RESTORE
```

```
                                0000    332          .SBTTL  - HIR Error Processing Information
                                0000    333
                                0000    334 ;  Constants used in forming HIR error messages
                                0000    335
                                0000    336          $VIELD  HIRER, 0, <-
                                0000    337                  <STEP,8,M>, -                  ; Step number
                                0000    338                  <TYPE,4,M>, -                  ; Error type
                                0000    339                  <ONLINE,,M> -                  ; Online (not HIR)
                                0000    340                  >
                     00000000   0000    341 HIRER$M_REPLACE = 0
                                0000    342
                                0000    343                                                 ; Error type codes:
                     00000001   0000    344 HIRER$K_READ    = 1                            ;       READ
                     00000002   0000    345 HIRER$K_WRITE   = 2                            ;       WRITE
                     00000003   0000    346 HIRER$K_RCTFULL = 3                            ;       RCT FULL
                     00000004   0000    347 HIRER$K_REPFAIL = 4                            ;       REPLACE FAILURE
                                0000    348
                                0000    349          .SAVE
                     00000000   0000    350          .PSECT  $$$301_HIR_ERRORS LONG
                                0000    351
                                0000    352 HIR_ERR_TYPES:
                                0000    353
                             00 0000    354          .BYTE   0
                 44 41 45 52 00' 0001    355          .ASCIC  /READ/
                             04 0001
              45 54 49 52 57 00' 0006    356          .ASCIC  /WRITE/
                             05 0006
        4C 4C 55 46 20 54 43 52 00' 000C    357          .ASCIC  /RCT FULL/
                             08 000C
49 41 46 20 45 43 41 4C 50 45 52 00' 0015    358          .ASCIC  /REPLACE FAILURE/
                    45 52 55 4C 0021
                             0F 0015
                                0025    359
                                0025    360 HIR_ERR_REPLACE:
                                0025    361
           45 43 41 4C 50 45 52 00' 0025    362          .ASCIC  /REPLACE/
                             07 0025
                                002D    363
                                002D    364 HIR_ERR_ONLINE:
                                002D    365
              45 4E 49 4C 4E 4F 00' 002D    366          .ASCIC  /ONLINE/
                             06 002D
                                0034    367
                                0034    368 HIR_ERR_SEG1:
                                0034    369
65 72 65 74 6E 75 6F 63 6E 65 20 00' 0034    370          .ASCIC  / encountered a /
                       20 61 20 64 0040
                             0F 0034
                                0044    371
                                0044    372 HIR_ERR_SEG2:
                                0044    373
   20 6E 69 20 72 6F 72 72 65 20 00' 0044    374          .ASCIC  / error in /
                             0A 0044
                                004F    375
                                004F    376 HIR_ERR_SEG3:
                                004F    377
              20 70 65 74 73 20 00' 004F    378          .ASCIC  / step /
```

```
                 06  004F
                     0056    379
                     0056    380 ;
                     0056    381 ; Compute maximum HIR message size
                     0056    382 ;
                     0056    383
00000012             0056    384 DEVNAMSIZ = 18                              ; size of a device name
00000007             0056    385 TYPSIZ    =  7                              ; largest error type character count
00000007             0056    386 FUNCSIZ   =  7                              ; largest REPLACE/ONLINE character count
00000002             0056    387 STEPSIZ   =  2                              ; max characters in step number
00000023             0056    388 FIXEDSIZ  = <. - HIR_ERR_SEG1> + 1          ; size of fixed text
                     0056    389
00000012             0056    390 HIRER$K_DEVNAMSIZ = DEVNAMSIZ
00000045             0056    391 SIZE = FIXEDSIZ + DEVNAMSIZ + TYPSIZ + FUNCSIZ + STEPSIZ
00000048             0056    392 HIRER$K_MSGSIZE = <SIZE + 3> & ^c3
                     0056    393
                 00000000    394         .RESTORE
                     0000    395
                     0000    396 ;++
                     0000    397 ;
                     0000    398 ; HIR_ERROR
                     0000    399 ;
                     0000    400 ;       This macro calls the HIR error reporting routine.
                     0000    401 ;
                     0000    402 ; Parameters:
                     0000    403 ;
                     0000    404 ;       STEP    step number in which the error occured
                     0000    405 ;       TYPE    error type (one of READ / WRITE / RCTFULL)
                     0000    406 ;       FUNC    function incuring error (one of REPLACE / ONLINE;
                     0000    407 ;               default = REPLACE)
                     0000    408 ;--
                     0000    409
                     0000    410         .MACRO  HIR_ERROR step, type, func=REPLACE
                     0000    411         ASSUME  HIRER$V_ONLINE LE 15
                     0000    412         MOVZWL  #<HIRER$M_'func' -
                     0000    413                 +<HIRER$K_'type' @ HIRER$V_TYPE > -
                     0000    414                 + 'step'>, R0
                     0000    415         BSBW    DU$HIR_ERROR
                     0000    416         .ENDM   HIR_ERROR
```

```
                                        .SBTTL  DU$INIT_HIRT - Initialize Host Initiated Replacement Table
                                ;++
                                ;
                                ; DU$INIT_HIRT - Initialize Host Initiated Replacement Table
                                ;
                                ; Functional Description:
                                ;
                                ;       This routine initializes the HIRT, if it has not already been
                                ;       initialized.  There is one HIRT per system and it resides in the
                                ;       disk class driver.  It is initialized the first time an intelligent
                                ;       controller that requires Host Initiated Bad Block Replacement is
                                ;       brought online.
                                ;
                                ;       HIRT initialization includes setting up its FLINK and BLINK, allocating
                                ;       a permanent CDRP for it, allocating an RSPID for it, allocating an
                                ;       MSCP buffer (without Send Credit on any connection) and allocating
                                ;       four pages of memory that are needed by the replacement algorithm.
                                ;
                                ; Inputs:
                                ;
                                ;       R3      CDDB address
                                ;       R4      PDT address
                                ;       R5      Connection permanent CDRP address
                                ;
                                ; Outputs:
                                ;
                                ;       Registers R0-R2 are modified.
                                ;       Registers R3-R5 are preserved.
                                ;
                                ; Implicit Outputs:
                                ;
                                ;       The HIRT is initialized as described above.
                                ;--
                                
                                DU$INIT_HIRT::
                                
 44 A3 8ED0  0000  454                  POPL    CDDB$L_SAVED_PC(R3)     ; Save caller's PC in CDDB.
    00   E3  0004  455                  BBCS    s^#HIRT$V_ACTIVE,-      ; Now see if HIRT already init'ed.
03 000E'CF      0006  456                  HIRT$W_STS,10$
    00A4   31  000A  457                  BRW     END_INIT_HIRT           ; Branch around if already initialized.
                000D  458
 0000'CF   D4  000D  459 10$:    CLRL    HIRT$L_RPLQFL           ; Singly linked list with second
 0000'CF   9E  0011  460                MOVAB   HIRT$L_RPLQFL,-         ;  longword pointing to tail of list.
 0004'CF         0015  461                  HIRT$L_RPLQTP
                0018  462
000E'CF   02   A8  0018  463                BISW    s^#HIRT$M_BUSY, -       ; Prevent use of HIRT until fully
                001D  464                  HIRT$W_STS              ;  init'ed.
                001D  465 ;
                001D  466 ; Allocate the CDRP to be used and re-used during the I/O operations
                001D  467 ; associated with dynamic Host Initiated Replacement of bad blocks.
                001D  468 ;
                001D  469
51  00C4 8F   3C  001D  470 20$:    MOVZWL  #IRP$K_LENGTH,R1        ; R1 contains amount of space to alloc.
    008F   30   0022  471                  BSBW    ALLOC_POOL              ; Allocate space. Returns R2=>space.
                0025  472
      0A   90   0025  473                  MOVB    #DYN$C_IRP,-            ; Make first part of CDRP look like an
   0A A2        0027  474                  IRP$B_TYPE(R2)          ;  IRP.
```

```
                                                     ;  Save type and size inside "IRP".
                                                     ;  R5 => CDRP portion of packet.
                                                     ;  Save address of replacement CDRP.
                                                     ;  Size field in CDRP portion is negative
                                                     ;   offset of IRP from CDRP portion.
                                                     ;  Mark type of CDRP portion.
                                                     ;  So far we have no connection for CDRP.
                                                     ;  This CDRP will not use RWAITCNT.
                                                     ;  Signal that no mapping resources allocated
                                                     ;  Clear RSPID to show none yet allocated.
                                                     ;  Likewise show no MSCP buffer.
                                                     ;  Set HIRT permanent CDRP flag.
                                                     ;
                                                     ; Allocate pages from pool to serve as buffers when reading RCT sectors
                                                     ; during replacement of bad blocks on a disk.
                                                     ;
51   020C 8F   3C   0053   496 50$:    MOVZWL  #512+12,R1                ;  R1 contains amount of space for a
                    0058   497                                         ;   page and a VMS structure header.
                    0058   498
55   00D0 C3   9E   0058   499         MOVAB   CDDB$A_PRMCDRP(R3), R5    ;  ALLOC_POOL needs R5 => Permanent CDRP.
        0054   30   005D   500         BSBW    ALLOC_POOL                ;  Allocate space. Returns R2=>space.
55   0020'CF   D0   0060   501         MOVL    HIRT$L_CDRP,R5            ;  Restore R5 => Hirt CDRP.
                    0065   502
                    0065   503         ASSUME  CDDB$B_SUBTYPE  EQ  CDDB$B_TYPE+1
              B0    0065   504         MOVW    #DYN$C_CLASSDRV-          ;  Place type and subtype descriptors
                    0066   505                 !<DYN$C_CD_BBRPG@8>,-     ;   into header using convenient (CDDB)
 0A A2    0264 8F        0066   506                 CDDB$B_TYPE(R2)           ;   offset definition.
       08 A2    51   B0   006B   507         MOVW    R1,CDDB$W_SIZE(R2)        ;  Also place size into header.
       52    0C A2   9E   006F   508         MOVAB   12(R2),R2                 ;  R2 => beyond VMS structure header.
             51   D4   0073   509         CLRL    R1                        ;  Clear loop index register.
                    0075   510
          0024'CF41   D5   0075   511 80$:    TSTL    HIRT$L_PAGEOPTR[R1]       ;  Test where to put address of allocated pag
                04   13   007A   512         BEQL    90$                       ;  EQL implies we have found a depository.
                51   D6   007C   513         INCL    R1                        ;  Else bump index register
                F5   11   007E   514         BRB     80$                       ;   and go back and try again.
                    0080   515
          0024'CF41   52   D0   0080   516 90$:    MOVL    R2,HIRT$L_PAGEOPTR[R1]    ;  Else save Page address.
0044'CF41    52   FE00 8F   AB   0086   517         BICW3   #^XFE00,R2,-       ;  Calculate BOFF of page just allocated
                    008F   518                 HIRT$W_BOFF0[R1]          ;   and save it in the indexed slot.
             09   EF   008F   519         EXTZV   S^#VA$V_VPN,-             ;  Now calculate SVAPTE of allocated
       52    52   15        0091   520                 S^#VA$S_VPN,R2,R2         ;   page.  First get VPN.
    50   00000000'GF   D0   0094   521         MOVL    G^MMG$GL_SPTBASE,R0       ;  Then R0 => base of system page table.
                    0098   522
 0034'CF41    6042   DE   0098   523         MOVAL   (R0)[R2],HIRT$L_SVAPTE0[R1]
                    00A2   524                                               ;  Move SVAPTE into proper slot.
          03   51   D1   00A2   525         CMPL    R1,#3                     ;  See if we are done allocating.
               AC   19   00A5   526         BLSS    50$                       ;  LSS implies NO, so we go to try again.
                    00A7   527
     000E'CF    02   AA   00A7   528         BICW    s^#HIRT$M_BUSY, -         ;  Allow use of HIRT now that it has
                    00AC   529                 HIRT$W_STS                ;   been initialized.
       55   00D0 C3   9E   00AC   530         MOVAB   CDDB$A_PRMCDRP(R3), R5    ;  Get controller permanent CDRP in R5.
                    00B1   531
```

```
                        00B1    532 END_INIT_HIRT:
 44 B3   17             00B1    533          JMP     @CDDB$L_SAVED_PC(R3)    ; Return to caller.
```

```
                                00B4    535          .SBTTL  ALLOC_POOL
                                00B4    536
                                00B4    537 ;+
                                00B4    538 ; This subroutine allocates and zeroes nonpaged pool.
                                00B4    539 ;
                                00B4    540 ; Inputs:
                                00B4    541 ;
                                00B4    542 ;       R1                      -# bytes of pool to allocate
                                00B4    543 ;       R5                      -Addr of CDRP
                                00B4    544 ;
                                00B4    545 ; Outputs:
                                00B4    546 ;
                                00B4    547 ;       R0                      -0/1 for fail/success
                                00B4    548 ;       R1                      -# bytes actually allocated
                                00B4    549 ;       R2                      -Addr of buffer allocated
                                00B4    550 ;-
                                00B4    551
                                00B4    552 ALLOC_POOL:                             ; Allocate and zero pool
                                00B4    553
                          53    DD    00B4    554          PUSHL   R3                 ; Save R3.
                00000000'GF     16    00B6    555          JSB     G^EXE$ALONONPAGED  ; Allocate from nonpaged pool
                          0E    50    E9    00BC    556          BLBC    R0,10$             ; Skip clearing structure if failure
                          37    BB    00BF    557          PUSHR   #^M<R0,R1,R2,R4,R5>; Save MOVC registers
    62    51    00    6E    00    2C    00C1    558          MOVC5   #0,(SP),#0,R1,(R2) ; Zero initialize structure
                                00C7    559
                          37    BA    00C7    560          POPR    #^M<R0,R1,R2,R4,R5>; Restore MOVC registers
                          53  8ED0    00C9    561          POPL    R3                 ; Restore R3.
                              05    00CC    562          RSB
                                00CD    563
                                00CD    564 10$:     ; Allocation failure.
                                00CD    565          ; Prepare to wait awhile before trying again.
                                00CD    566
                      10  A5  8ED0    00CD    567          POPL    CDRP$L_FR3(R5)             ; Save R3 in R5=>UCB or CDRP.
                14    A5    54    D0    00D1    568          MOVL    R4, CDRP$L_FR4(R5)         ; Likewise R4
                      18  A5  8ED0    00D5    569          POPL    CDRP$L_SAVD_RTN(R5)        ; and caller's return address.
                20    A5    51    D0    00D9    570          MOVL    R1, CDRP$L_RSPID(R5)       ; Save allocation size.
                                00DD    571          FORK_WAIT                          ; Wait awhile.
                51    20  A5    D0    00E3    572          MOVL    CDRP$L_RSPID(R5), R1       ; Restore size of block to allocate.
                      20  A5    D4    00E7    573          CLRL    CDRP$L_RSPID(R5)           ; Restore CDRP field.
                      18  A5    DD    00EA    574          PUSHL   CDRP$L_SAVD_RTN(R5)        ; Restore caller's return address.
                          C5    11    00ED    575          BRB     ALLOC_POOL                 ; Go try again.
```

```
                     00EF    577          .SBTTL  DU$LOCK_MIRT - Gain exclusive access to MIRT
                     00EF    578 ;++
                     00EF    579 ;
                     00EF    580 ; DU$LOCK_MIRT - Gain exclusive access to MIRT
                     00EF    581 ;
                     00EF    582 ; Functional Description:
                     00EF    583 ;
                     00EF    584 ;       Gain exclusive access to the H(ost) I(initiated) bad block
                     00EF    585 ;       R(eplacement) T(able).  Also, lockout new activity on the unit by
                     00EF    586 ;       bumping UCB$W_RWAITCNT.
                     00EF    587 ;
                     00EF    588 ; Inputs:
                     00EF    589 ;
                     00EF    590 ;       R3      UCB address
                     00EF    591 ;       R5      a fork block
                     00EF    592 ;
                     00EF    593 ; Implicit Inputs:
                     00EF    594 ;
                     00EF    595 ;        (SP)   caller's return address
                     00EF    596 ;       4(SP)   caller's caller's return address
                     00EF    597 ;
                     00EF    598 ; Outputs:
                     00EF    599 ;
                     00EF    600 ;       R0 - R2 are modified.
                     00EF    601 ;       All other registers are preserved.
                     00EF    602 ;
                     00EF    603 ; Implicit Outputs:
                     00EF    604 ;
                     00EF    605 ;       MIRT is owned by caller.
                     00EF    606 ;--
                     00EF    607
                     00EF    608 DU$LOCK_MIRT::
                     00EF    609
000E'CF   00   E0    00EF    610          BBS     s^#MIRT$V_ACTIVE, -     ; Make sure MIRT is active.
          04                 00F4    611                  MIRT$W_STS, 5$
                     00F5    612          BUG_CHECK       DISKCLASS,FATAL
                     00F9    613
      56  A3   B6    00F9    614 5$:      INCW    UCB$W_RWAITCNT(R3)      ; Prevent new CDRP's from being begun
                     00FC    615                                          ;  on this UCB.
000E'CF   01   E3    00FC    616          BBCS    s^#MIRT$V_BUSY, -               ; Allocate the Host Initiated Replac
          1B                 0101    617                  MIRT$W_STS, 10$         ;  ment table (MIRT).
                     0102    618
   10 A5  53   7D    0102    619          MOVQ    R3, FKB$L_FR3(R5)       ; If here, already allocated, save
      0C  A5 8ED0    0106    620          POPL    FKB$L_FPC(R5)           ;  thread context in fork block.
                     010A    621
                     010A    622          ; Thread R5 (a fork block) onto the tail of the singly threaded list
                     010A    623          ; of fork blocks awaiting use of the MIRT.  The listhead is a quadword
                     010A    624          ; whose first longword points to the first fork block on the list and
                     010A    625          ; whose second longword points to the last forkblock on the list.  An
                     010A    626          ; empty list is characterized by having the first longword contain a
                     010A    627          ; zero with the second longword pointing to the first longword.  Each
                     010A    628          ; fork block on the list, has the first longword of its link quadword
                     010A    629          ; pointing to the next fork block on the list, with the last fork
                     010A    630          ; block containing a zero in this longword.  The second longword of
                     010A    631          ; each fork block's link quadword contains the address of the CDDB of
                     010A    632          ; the intelligent controller associated with the device unit
                     010A    633          ; attempting to gain exclusive use of the MIRT.
```

```
                          010A   634          ;
                          010A   635          ; Note the reason for CDDB address here is to facilitate finding CDRPs
                          010A   636          ; associated with a CONNECTION that has failed (gone down).
                          010A   637          ;
                65   D4   010A   638          CLRL    FKB$L_FQFL(R5)          ; Prepare this fork block to be at
                          010C   639          ;  tail of the list.
    04 A5   00BC C3   D0   010C   640          MOVL    UCB$L_CDDB(R3), -       ; Second longword of link quadword
                          0112   641                  FKB$L_FQBL(R5)          ;  points to CDDB.
       0004'DF  65   9E   0112   642          MOVAB   FKB$L_FQFL(R5), -       ; Move address of this fork block into
                          0117   643                  @HIRT$L_RPLQTP          ;  forward ptr of previous tail.
       0004'CF  65   9E   0117   644          MOVAB   FKB$L_FQFL(R5), -       ; Also move address of this fork block
                          011C   645                  HIRT$L_RPLQTP           ;  to list tail pointer.
                05   011C   646          RSB                             ; Terminate this execution thread by
                          011D   647                                          ;  returning to caller's caller.
                          011D   648          ;
                          011D   649 10$:     ; The HIRT is owned.
                01   10   011D   650          BSBB    GRANT_HIRT              ; Call to initialize various structures
                          011F   651                                          ;  with data of the new HIRT owner.
                05   011F   652          RSB                             ; And return to caller who now owns HIRT.
```

```
                     0120   654          .SBTTL  GRANT_HIRT - Complete granting access to the HIRT
                     0120   655 ;++
                     0120   656 ;
                     0120   657 ; GRANT_HIRT - Complete granting access to the HIRT
                     0120   658 ;
                     0120   659 ; Functional Description:
                     0120   660 ;
                     0120   661 ;       This routine is called from DU$LOCK_HIRT and DU$UNLOCK_HIRT, upon
                     0120   662 ;       granting ownership of the HIRT to a thread.  GRANT_HIRT initializes
                     0120   663 ;       various data fields reflecting this ownership and facilitating the
                     0120   664 ;       thread's use of the HIRT CDRP.
                     0120   665 ;
                     0120   666 ; Note:
                     0120   667 ;       Since both subroutines that require ownership of the HIRT, REPLACE_LBN
                     0120   668 ;       and ONLINE_COMPLETE, make use of the user's original RSPID so as to be
                     0120   669 ;       able to co-relate all Error Log messages generated by a user I/O
                     0120   670 ;       request, GRANT_HIRT passes the RSPID form the user CDRP to the HIRT
                     0120   671 ;       permanent CDRP.
                     0120   672 ;
                     0120   673 ; Inputs:
                     0120   674 ;
                     0120   675 ;       R3      UCB address
                     0120   676 ;       R5      User CDRP address
                     0120   677 ;
                     0120   678 ;Outputs:
                     0120   679 ;
                     0120   680 ;       Various HIRT and CDRP fields updated.
                     0120   681 ;--
                     0120   682
                     0120   683 GRANT_HIRT:
                     0120   684
           20 A5   DD  0120   685          PUSHL   CDRP$L_RSPID(R5)        ; Pass current RSPID to HIRT CDRP.
           20 A5   D4  0123   686          CLRL    CDRP$L_RSPID(R5)        ; Prevent spurious deallocates.
     001C'CF    55   D0  0126   687          MOVL    R5, HIRT$L_SAVDCDRP     ; Save given R5.
0070'CF  0074'CF   9E  012B   688          MOVAB   HIRT$L_SUBSTACK, -      ; Initialize SUBSTACK in HIRT.
                     0132   689                  HIRT$L_STKPTR
     0014'CF    53   D0  0132   690          MOVL    R3, HIRT$L_OWNUCB       ; Indicate who owns HIRT.
                     0137   691
          55    6E   D0  0137   692          MOVL    (SP), R5                ; Get RSPID.
          7E    50   7D  013A   693          MOVQ    R0, -(SP)               ; Save registers.
                     013D   694          FIND_RSPID_RDTE                 ; Lookup RDT for RSPID.
            04  50   E8  0143   695          BLBS    R0, 10$                 ; Branch if lookup successful.
                     0146   696          BUG_CHECK       DISKCLASS,FATAL ; Else, major inconsistancy.
     65   0020'CF   D0  014A   697 10$:       MOVL    HIRT$L_CDRP, -          ; For now pass ownership of RDTE to HIRT
                     014F   698                  RD$L_CDRP(R5)           ;  permanent CDRP.
          50   8E   7D  014F   699          MOVQ    (SP)+, R0               ; Restore saved registers.
     55   0020'CF   D0  0152   700          MOVL    HIRT$L_CDRP,R5          ; R5 => permanent replacement CDRP.
            20 A5 8ED0  0157   701          POPL    CDRP$L_RSPID(R5)        ; Pickup RSPID to use thruout replacement.
            2C A5   D4  015B   702          CLRL    CDRP$L_LBUFH_AD(R5)     ; Indicate no resources yet allocated
            1C A5   D4  015E   703          CLRL    CDRP$L_MSG_BUF(R5)      ;  except RSPID.
        BC A5   53   D0  0161   704          MOVL    R3,CDRP$L_UCB(R5)       ; Make HIRT permanent CDRP => this UCB.
                     0165   705                                          ;  This allow UNIBUS mapping to work.
          00C8 C3   D0  0165   706          MOVL    UCB$L_CDT(R3),-         ; Place CDT pointer into CDRP for handy
            24 A5       0169   707                  CDRP$L_CDT(R5)          ;  reference by SCS routines.  Note this
                     016B   708                                          ;  must be done each time the HIRT is
                     016B   709                                          ;  locked since we may be using a different
                     016B   710                                          ;  port (and therefore CONNECTION) each
```

```
                        016B  711
                              ;  time.
          56 A3   9E    016B  712          MOVAB   UCB$W_RWAITCNT(R3),-     ; Point CDRP field to UCB field.
          28 A5         016E  713                  CDRP$L_RWCPTR(R5)
    0080 8F       AA  0170  714          BICW    #HIRT$M_ERLOGIP,-         ; Initialize bit.
    000E'CF             0174  715                  HIRT$W_STS
                  05    0177  716          RSB                               ; Return to caller.
```

```
                     0178   718          .SBTTL  DU$UNLOCK_HIRT - Release HIRT access
                     0178   719 ;++
                     0178   720 ;
                     0178   721 ; DU$UNLOCK_HIRT - Release HIRT access
                     0178   722 ;
                     0178   723 ; Functional Description:
                     0178   724 ;
                     0178   725 ;       Caller wishes to relinquish exclusive control of the HIRT.
                     0178   726 ;       It becomes the current owner's obligation to restart the first
                     0178   727 ;       thread (if any are there) that may be waiting on the HIRT wait
                     0178   728 ;       list.
                     0178   729 ;
                     0178   730 ; Note:
                     0178   731 ;
                     0178   732 ;       DU$UNLOCK_HIRT passes back the user's RSPID from the HIRT permanent
                     0178   733 ;       CDRP to the user's CDRP.
                     0178   734 ;
                     0178   735 ; Inputs:
                     0178   736 ;
                     0178   737 ;       R3      UCB of HIRT owner
                     0178   738 ;
                     0178   739 ; Implicit Inputs:
                     0178   740 ;
                     0178   741 ;       HIRT owned by caller
                     0178   742 ;
                     0178   743 ; Outputs:
                     0178   744 ;
                     0178   745 ;       R5      Original CDRP address
                     0178   746 ;       All other registers are preserved.
                     0178   747 ;
                     0178   748 ; Implicit Outputs:
                     0178   749 ;
                     0178   750 ;       HIRT ownership relinquished.  If any threads are on the HIRT wait list,
                     0178   751 ;       first of these is granted HIRT ownership and is started up.
                     0178   752 ;--
                     0178   753
                     0178   754 DU$UNLOCK_HIRT::
                     0178   755
      56 A3    B7    0178   756          DECW    UCB$W_RWAITCNT(R3)      ; Decrement to again allow normal I/O.
    7E   50    7D    017B   757          MOVQ    R0, -(SP)               ; Save some registers.
      1C   BB    017E   758          PUSHR   #^M<R2,R3,R4>           ; Save more registers.
    55   53    D0    0180   759          MOVL    R3, R5                  ; Setup UCB for UNSTALLUCB.
00000000'GF   16    0183   760          JSB     G^$CS$UNSTALLUCB        ; Call to start up IRP's on UCB$L_IOQFL.
      1C   BA    0189   761          POPR    #^M<R2,R3,R4>           ; Restore registers.
                     018B   762
55  0020'CF    D0    018B   763          MOVL    HIRT$L_CDRP,R5          ; R5 => HIRT CDRP.
      20 A5    DD    0190   764          PUSHL   CDRP$L_RSPID(R5)        ; Save current RSPID so as to restore to
                     0193   765                                          ;  user CDRP.
      2C   13    0193   766          BEQL    15$                     ; EQL implies RSPID has been deallocated
                     0195   767                                          ;  due to re-CONNECT. Branch around.
      20 A5    D4    0195   768          CLRL    CDRP$L_RSPID(R5)        ; Prevent spurious deallocates.
    55   6E    D0    0198   769          MOVL    (SP), R5                ; Get RSPID.
                     019B   770          FIND_RSPID_RDTE                 ; Lookup RDT entry for RSPID.
      04 50    E8    01A1   771          BLBS    R0, 5$                  ; Branch if lookup successful.
                     01A4   772          BUG_CHECK       DISKCLASS,FATAL ; Else, major inconsistancy.
50  001C'CF    D0    01A8   773 5$:       MOVL    HIRT$L_SAVDCRP, R0      ; Get saved CDRP address.
      0F   12    01AD   774          BNEQ    10$                     ; Branch if there still is a saved CDRP.
```

```
                     01AF   775                                    ; Else, it has been canceled.
          55   65    D0  01AF   776          MOVL    RD$L_CDRP(R5), R5       ; Which means, use HIRT CDRP.
            20 A5 8ED0  01B2   777          POPL    CDRP$L_RSPID(R5)        ; Restore its RSPID so it can be
                     01B6   778          DEALLOC_RSPID                   ; deallocated.
               11   11  01BC   779          BRB     19$                     ; And branch around.
                     01BE   780
          65   50    D0  01BE   781 10$:     MOVL    R0, RD$L_CDRP(R5)       ; Pass ownership of RDTE back to user
               50 8ED0  01C1   782 15$:     POPL    R0                      ; Get RSPID.
     55   001C'CF    D0  01C4   783          MOVL    HIRT$L_SAVDCDRP, R5     ; Get original CDRP address in R5.
               04   13  01C9   784          BEQL    19$                     ; Branch if original CDRP canceled.
       20 A5   50    D0  01CB   785          MOVL    R0, CDRP$L_RSPID(R5)    ; Else, restore user's original RSPID.
                     01CF   786
          50   8E   7D  01CF   787 19$:     MOVQ    (SP)+, R0               ; Restore 1st group of saved registers.
                     01D2   788
                     01D2   789
          0000'CF    D5  01D2   790          TSTL    HIRT$L_RPLQFL           ; Determine if HIRT wait list is empty.
               27   13  01D6   791          BEQL    50$                     ; EQL implies list empty.
                     01D8   792
               3F   BB  01D8   793          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save caller's registers.
                     01DA   794
     55   0000'CF    D0  01DA   795          MOVL    HIRT$L_RPLQFL,R5        ; R5 => 1st fork block on list.
               65   D0  01DF   796          MOVL    FKB$L_FQFL(R5),-        ; Replace 1st fork block on list with
          0000'CF         01E1   797                  HIRT$L_RPLQFL           ;  next fork block.
               07   12  01E4   798          BNEQ    35$                     ; NEQ implies there was a next fork block.
          0000'CF   9E  01E6   799          MOVAB   HIRT$L_RPLQFL,-         ; Else wait list is now empty, so re-
          0004'CF         01EA   800                  HIRT$L_RPLQTP           ;  direct list Tail Pointer to listhead.
       53   10 A5   7D  01ED   801 35$:     MOVQ    FKB$L_FR3(R5),R3        ; Restore waiting thread's context.
             FF2C   30  01F1   802          BSBW    GRANT_HIRT              ; Call to initialize various structures
                     01F4   803                                          ;  with data of the new HIRT owner.
     50   001C'CF    D0  01F4   804          MOVL    HIRT$L_SAVDCDRP,R0      ; R0 => User CDRP. Now
             0C B0   16  01F9   805          JSB     @FKB$L_FPC(R0)          ;  resume its waiting thread.
               3F   BA  01FC   806          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore relinquisher's registers.
                     05  01FE   807          RSB                             ; And return to relinquisher.
                     01FF   808
                     01FF   809 50$:                                      ; If here, list was empty.
     000E'CF    02   AA  01FF   810          BICW    s^#HIRT$M_BUSY, -       ; So mark HIRT as NOT busy.
                     0204   811                  HIRT$W_STS
                     05  0204   812          RSB                             ; And return to relinquisher.
```

```
                     0205    814          .SBTTL  DU$TEST_HIRT_RWAITCNT - Accumulate RWAITCNT for HIRT
                     0205    815 ;++
                     0205    816 ;
                     0205    817 ; DU$TEST_HIRT_RWAITCNT - Accumulate RWAITCNT for HIRT
                     0205    818 ;
                     0205    819 ; Functional Description:
                     0205    820 ;
                     0205    821 ;       This routine accumulates an RWAITCNT value for the input UCB based
                     0205    822 ;       upon the amount RWAITCNT has been increment for HIRT usage.
                     0205    823 ;
                     0205    824 ; Inputs:
                     0205    825 ;
                     0205    826 ;       R0      RWAITCNT accumulator
                     0205    827 ;       R5      UCB address
                     0205    828 ;
                     0205    829 ; Outputs:
                     0205    830 ;
                     0205    831 ;       R0      RWAITCNT accumulator (with additions for HIRT usage)
                     0205    832 ;       R1      destroyed
                     0205    833 ;
                     0205    834 ;       All other registers preserved.
                     0205    835 ;--
                     0205    836
                     0205    837 DU$TEST_HIRT_RWAITCNT::
                     0205    838
1D 000E'CF   01   E1 0205    839          BBC     s^#HIRT$V_BUSY, HIRT$W_STS, 90$ ; Branch if HIRT not busy.
                     0208    840
   0014'CF   55   D1 0208    841          CMPL    R5, HIRT$L_OWNUCB               ; Is the UCB the HIRT owner?
             02   12 0210    842          BNEQ    10$                             ; Branch if not HIRT owner.
             50   D6 0212    843          INCL    R0                              ; Else, increment RWAITCNT.
                     0214    844
                     0214    845 10$:     ASSUME  FKB$L_FQFL EQ 0
51   0000'CF   9E   0214    846          MOVAB   HIRT$L_RPLQFL, R1               ; Init. "previous" wait CDRP.
     51   61   D0 0219    847 11$:     MOVL    FKB$L_FQFL(R1), R1              ; Link to next waiting CDRP.
          0A   13 021C    848          BEQL    90$                             ; Branch if no more waiters.
  BC A1   55   D1 021E    849          CMPL    R5, CDRP$L_UCB(R1)              ; Is this waiter for this UCB?
          F5   12 0222    850          BNEQ    11$                             ; Branch if not right UCB.
          50   D6 0224    851          INCL    R0                              ; Else, increment RWAITCNT.
          F1   11 0226    852          BRB     11$                             ; Loop, till no more waiters.
                     0228    853
                  05 0228    854 90$:     RSB                                     ; All done; exit.
```

```
                     0229   856          .SBTTL  DU$CANCEL_FROM_HIRT - Cancel requests from the HIRT
                     0229   857 ;++
                     0229   858 ;
                     0229   859 ; DU$CANCEL_FROM_HIRT - Cancel requests from the HIRT
                     0229   860 ;
                     0229   861 ; Functional Description:
                     0229   862 ;
                     0229   863 ;       This routine is called to locate and cancel any I/O requests current
                     0229   864 ;       or pending for host initiated replacement.  The queue of pending
                     0229   865 ;       requests is scanned.  The then current HIRT owner is tested.
                     0229   866 ;
                     0229   867 ;       The HIRT wait queue is scanned and all CDRPs that meet the cancel
                     0229   868 ;       criteria are removed from the HIRT wait queue and queued for I/O post
                     0229   869 ;       processing.  The current owner of the HIRT (if any) is similarly
                     0229   870 ;       tested against the cancel criteria and if needed it too is queued for
                     0229   871 ;       I/O post processing.  The HIRT is left "ownerless" in the sense that
                     0229   872 ;       HIRT$L_SAVDCDRP is left zero.  This allows the current HIRT I/O to
                     0229   873 ;       continue until it completes on its own.  Then, when the HIRT is
                     0229   874 ;       UNLOCKED, the "ownerless" state is noticed and the HIRT thread for the
                     0229   875 ;       former owner is evaporated.
                     0229   876 ;
                     0229   877 ; Inputs:
                     0229   878 ;
                     0229   879 ;       R3      UCB address
                     0229   880 ;       R5      Cancel CDRP address
                     0229   881 ;
                     0229   882 ; Implicit Inputs:
                     0229   883 ;
                     0229   884 ;       CDRP$W_DUTUCNTR(R5)     count of number of times to increment RWAITCNT
                     0229   885 ;                               after cancel is completed.
                     0229   886 ;
                     0229   887 ; Outputs:
                     0229   888 ;
                     0229   889 ;       R0 through R2 are destroyed
                     0229   890 ;       All other registers are preserved.
                     0229   891 ;
                     0229   892 ; Implicit Outputs:
                     0229   893 ;
                     0229   894 ;       CDRP$W_DUTUCNTR(R5)     count of number of times to increment RWAITCNT
                     0229   895 ;                               after cancel is completed.
                     0229   896 ;--
                     0229   897
                     0229   898 DU$CANCEL_FROM_HIRT::
                     0229   899
000E'CF   01   E1    0229   900          BBC     s^#HIRT$V_BUSY, -       ; Is the HIRT busy?  If not, there is
                  79 022E   901                  HIRT$W_STS, 900$        ; nothing to do: so branch to exit.
                     022F   902
                     022F   903          ; Scan the HIRT pending requests queue
                     022F   904
                     022F   905          ASSUME  FKB$L_FQFL EQ 0
51   0000'CF   9E    022F   906          MOVAB   HIRT$L_RPLQFL, R1       ; Get "previous" CDRP on wait list.
                     0234   907
      52   61   D0  0234   908 10$:     MOVL    FKB$L_FQFL(R1), R2      ; Get nect CDRP.
                  3C 13 0237   909          BEQL    100$                    ; Branch if no more CDRPs on wait list.
04 A2   00BC C3   D1 0239   910          CMPL    UCB$L_CDDB(R3), -       ; Is CDRP for this CDDB?
                     023F   911                  FKB$L_FQBL(R2)
                  06 12 023F   912          BNEQ    40$                     ; Branch if not the right CDDB.
```

```
                     0241   913                  IFCANCEL cdrp=(R2), then=70$     ; Branch if CDRP should be canceled.
          51   52    D0 0247   914 40$:          MOVL    R2, R1                   ; Current becomes previous.
               E8    11 024A   915               BRB     10$                      ; Loop through all waiting CDRPs.
                     024C   916
          61   62    D0 024C   917 70$:          MOVL    FKB$L_FQFL(R2), -        ; Unlink cancelable CDRP.
                     024F   918                          FKB$L_FQFL(R1)
               05    12 024F   919               BNEQ    75$                      ; If cancelable CDRP was last,
    0004'CF    51    D0 0251   920               MOVL    R1, HIRT$L_RPLQTP        ; adjust queue tail pointer.
          50   52    D0 0256   921 75$:          MOVL    R2, R0                   ; Setup CDRP to cancel.
            44 A5    B6 0259   922               INCW    CDRP$W_DUTUCNTR(R5)      ; Account for RWAITCNT increment during
                     025C   923                                                   ; attempt to lock the HIRT.
               3F    BB 025C   924               PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; Save registers.
          55   50    D0 025E   925               MOVL    R0, R5                   ; Setup for message deallocate.
                     0261   926               DEALLOC_MSG_BUF                  ; Deallocate End Message that told of
                     0264   927                                                   ; block to be replaced.
               3F    BA 0264   928               POPR    #^M<R0,R1,R2,R3,R4,R5>   ; Restore registers.
                     0266   929               POST_CDRP status=SS$_CANCEL      ; Insert IRP/CDRP in IOPOST queue.
               BF    11 0273   930               BRB     10$                      ; Branch back to scan entire list.
                     0275   931
                     0275   932 100$:    ; Is the HIRT owner a cancelable CDRP?  If so retrieve this HIRT owner
                     0275   933          ; CDRP, clear HIRT$L_SAVDCDRP, and POST_CDRP the retrieved CDRP.  Note
                     0275   934          ; this works in conjunction with DU$UNLOCK_HIRT and DU$RSTRTQ_HIRT_CDRP
                     0275   935          ; which must be prepared to find HIRT$L_SAVDCDRP = 0.
                     0275   936
    BC A5    0014'CF    D1 0275   937               CMPL    HIRT$L_OWNUCB, -         ; Check for correct HIRT owner UCB.
                     027B   938                          CDRP$L_UCB(R5)
               2B    12 027B   939               BNEQ    900$                     ; Branch in wrong HIRT owner.
          52   001C'CF    D0 027D   940               MOVL    HIRT$L_SAVDCDRP, R2      ; Get CDRP owner of HIRT.
               24    13 0282   941               BEQL    900$                     ; Branch if owner already canceled,
                     0284   942                                                   ; replacement running to completion.
                     0284   943               IFNOCANCEL cdrp=(R2), then=900$ ; Branch if owner shouldn't be canceled.
    001C'CF    D4 028A   944               CLRL    HIRT$L_SAVDCDRP          ; Else, indicate HIRT owner canceled.
    50   52    D0 028E   945               MOVL    R2, R0                   ; Setup CDRP to cancel.
                     0291   946          ; Following instruction deleted due to its causing RWAITCNT to be
                     0291   947          ; decremented twice; once here and once Replacement runs to completion.
                     0291   948          ;INCW    CDRP$W_DUTUCNTR(R5)      ; Account for owning the HIRT.
               3F    BB 0291   949               PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; Save registers.
    55   50    D0 0293   950               MOVL    R0, R5                   ; Setup for message deallocate.
                     0296   951               DEALLOC_MSG_BUF                  ; Deallocate End Message that told of
                     0299   952                                                   ; block to be replaced.
               3F    BA 0299   953               POPR    #^M<R0,R1,R2,R3,R4,R5>   ; Restore registers.
                     029B   954               POST_CDRP status=SS$_CANCEL      ; Insert IRP/CDRP in IOPOST queue.
                     02A8   955
                05    05 02A8   956 900$:    RSB                              ; Return to caller.
```

```
                                02A9   958          .SBTTL  DU$DISCONNECT_HIRT - Do HIRT cleanup for a disconnect
                                02A9   959 ;++
                                02A9   960 ;
                                02A9   961 ; DU$DISCONNECT_HIRT - Do HIRT cleanup for a disconnect
                                02A9   962 ;
                                02A9   963 ; Functional Description:
                                02A9   964 ;
                                02A9   965 ;       Scan the HIRT wait queue for CDRPs belonging to this CDDB.  Remove
                                02A9   966 ;       them and place on the restart queue.  This must be done before the RDT
                                02A9   967 ;       resource wait is scanned.  It is essential to deallocate SCS resources
                                02A9   968 ;       held by CDRPs on the HIRT wait queue before scanning any of the SCS
                                02A9   969 ;       resource wait queues.
                                02A9   970 ;
                                02A9   971 ; Inputs:
                                02A9   972 ;
                                02A9   973 ;       R3      CDDB address
                                02A9   974 ;       R4      PDT address
                                02A9   975 ;       R5      Permanent CDRP address
                                02A9   976 ;
                                02A9   977 ; Outputs:
                                02A9   978 ;
                                02A9   979 ;       R0 through R2 are destroyed.
                                02A9   980 ;       All other registers are preserved.
                                02A9   981 ;--
                                02A9   982
                                02A9   983 DU$DISCONNECT_HIRT::
                                02A9   984
000E'CF   00   E1               02A9   985          BBC     s^#HIRT$V_ACTIVE, -     ; See if HIRT has been activated.
                  30            02AE   986                  HIRT$W_STS, 99$         ; If HIRT not active, branch around.
                  55   DD       02AF   987          PUSHL   R5                      ; Save a register.
                                02B1   988
                                02B1   989 30$:     ASSUME  FKB$L_FQFL EQ 0
50   0000'CF   9E               02B1   990          MOVAB   HIRT$L_RPLQFL, R0       ; Get "previous" CDRP on wait list.
                                02B6   991
     55   60   D0               02B6   992 40$:     MOVL    FKB$L_FQFL(R0), R5      ; Get next CDRP on wait list.
                  21   13       02B9   993          BEQL    90$                     ; Branch if no more waiting CDRPs.
  04 A5          53   D1       02BB   994          CMPL    R3, FKB$L_FQBL(R5)      ; See if waiter has right CDDB.
                  16   12       02BF   995          BNEQ    60$                     ; Branch if wrong CDDB.
     60   65   D0               02C1   996          MOVL    FKB$L_FQFL(R5), -       ; Let previous point to next.
                                02C4   997                  FKB$L_FQFL(R0)
                  05   12       02C4   998          BNEQ    50$                     ; Branch if current CDRP is not last.
0004'CF   50   D0               02C6   999          MOVL    R0, HIRT$L_RPLQTP       ; Else, previous is new end.
  50    BC   A5   D0            02CB  1000 50$:     MOVL    CDRP$L_UCB(R5), R0      ; Get UCB of interest.
        56   A0   B7            02CF  1001          DECW    UCB$W_RWAITCNT(R0)      ; Decrement count incremented during
                                02D2  1002                                          ; attempt to allocate HIRT.
           FD2B'   30           02D2  1003          BSBW    DUTU$INSERT_RESTARTQ    ; Insert this CDRP in restart queue.
                  DA   11       02D5  1004          BRB     30$                     ; Branch back to re-scan entire
                                02D7  1005                                          ; HIRT wait queue.
                                02D7  1006 60$:     ; Setup to move to next waiting CDRP.
     50   55   D0               02D7  1007          MOVL    R5, R0                  ; Current becomes previous.
                  DA   11       02DA  1008          BRB     40$                     ; Loop back.
                                02DC  1009
           55 8ED0              02DC  1010 90$:     POPL    R5                      ; Restore saved register.
                     05         02DF  1011 99$:     RSB                             ; Return to caller.
```

```
                     02E0  1013          .SBTTL  DU$RSTRTQ_HIRT_CDRP - Do connection failed cleanup of HIRT CDRP
                     02E0  1014 ;++
                     02E0  1015 ;
                     02E0  1016 ; DU$RSTRTQ_HIRT_CDRP - Do connection failed cleanup of HIRT CDRP
                     02E0  1017 ;
                     02E0  1018 ; Functional Description:
                     02E0  1019 ;
                     02E0  1020 ;       This routine is called by DUTU$INSERT_RESTARTQ when it is discovered
                     02E0  1021 ;       that the CDRP destined for the restart queue is the HIRT permanent
                     02E0  1022 ;       CDRP.  This action is taken instead of placing the HIRT permanent CDRP
                     02E0  1023 ;       on the restart queue.
                     02E0  1024 ;
                     02E0  1025 ;       The CDRP owning the HIRT is located and processed with a recursive
                     02E0  1026 ;       call to DUTU$INSERT_RESTARTQ.  Any mapping resources owned by the HIRT
                     02E0  1027 ;       permanent CDRP are copied to one of the CDDB permanent CDRPs.  This
                     02E0  1028 ;       allows the resources to be deallocated sometime after the connection
                     02E0  1029 ;       is DISCONNECTed.  This prevents "insane" servers for incorrectly
                     02E0  1030 ;       overwriting memory due to reallocation of mapping resources.  Finally,
                     02E0  1031 ;       the HIRT is unlocked, thus making it available for some other
                     02E0  1032 ;       replacement operation.
                     02E0  1033 ;
                     02E0  1034 ; Inputs:
                     02E0  1035 ;
                     02E0  1036 ;       R3      CDDB address
                     02E0  1037 ;       R4      PDT address
                     02E0  1038 ;       R5      HIRT permanent CDRP address
                     02E0  1039 ;
                     02E0  1040 ; Outputs:
                     02E0  1041 ;
                     02E0  1042 ;       R0 is destroyed.
                     02E0  1043 ;       All other registers are preserved.
                     02E0  1044 ;--
                     02E0  1045
                     02E0  1046 DU$RSTRTQ_HIRT_CDRP::
                     02E0  1047
          55    DD   02E0  1048          PUSHL   R5                              ; Save permanent replacement CDRP addr.
 55  001C'CF    D0   02E2  1049          MOVL    HIRT$L_SAVDCDRP, R5             ; Get HIRT owner CDRP address.
          08    13   02E7  1050          BEQL    10$                             ; Branch if HIRT owner was canceled.
03 40 A5    03    E0   02E9  1051          BBS     #CDRP$V_PERM, -                 ; Branch if HIRT owner was a CDDB
                   02EE  1052                  CDRP$L_DUTUFLAGS(R5), -         ; permanent CDRP.
                   02EE  1053                  10$
        FD0F'    30   02EE  1054          BSBW    DUTU$INSERT_RESTARTQ            ; Insert HIRT owner on restart queue.
          55  8ED0   02F1  1055 10$:     POPL    R5                              ; Restore HIRT permanent CDRP addr.
                   02F4  1056
        2C A5    D5   02F4  1057          TSTL    CDRP$L_LBUFH_AD(R5)             ; Were mapping resources allocated?
          17    13   02F7  1058          BEQL    20$                             ; Branch if no mapping res. allocated.
        2C A5    D4   02F9  1059          CLRL    CDRP$L_LBUFH_AD(R5)             ; Prevent duplicate deallocations.
 50  00D0 C3    9E   02FC  1060          MOVAB   CDDB$A_PRMCDRP(R3), R0          ; Get CDDB permanent CDRP address.
2C A0    30 A0    9E   0301  1061          MOVAB   CDRP$T_LBUFHNDL(R0), -          ; Put address of Local BUFfer HaNDLe
                   0306  1062                  CDRP$L_LBUFH_AD(R0)             ; field into field that points to it.
                   0306  1063          ASSUME  CDRP$L_UBARSRCE EQ CDRP$T_LBUFHNDL+12
30 A0    30 A5    7D   0306  1064          MOVQ    CDRP$T_LBUFHNDL(R5), -          ; Copy contents of buffer handle to
                   0308  1065                  CDRP$T_LBUFHNDL(R0)             ; CDDB permanent CDRP.  Also copy
38 A0    38 A5    7D   0308  1066          MOVQ    CDRP$T_LBUFHNDL+8(R5), -        ; CDRP$L_UBARSRCE in case this is
                   0310  1067                  CDRP$T_LBUFHNDL+8(R0)           ; a UNIBUS controller.
                   0310  1068
          53    DD   0310  1069 20$:     PUSHL   R3                              ; Save CDDB address.
```

```
                53   BC A5    D0  0312  1070          MOVL    CDRP$L_UCB(R5), R3      ; Setup UCB for unlocking HIRT.
                  FE5F    30  0316  1071          BSBW    DU$UNLOCK_HIRT          ; Release HIRT.
                     53 8ED0  0319  1072          POPL    R3                      ; Restore CCDB address.
                              031C  1073
                        05    031C  1074          RSB                             ; Return
```

```
                        031D  1076          .SBTTL  DU$REPLACE_LBN - Replace a failing block
                        031D  1077 ;++
                        031D  1078 ;
                        031D  1079 ; DU$REPLACE_LBN - Replace a failing block
                        031D  1080 ;
                        031D  1081 ; Functional Description:
                        031D  1082 ;
                        031D  1083 ;       Perform dynamic bad block replacement.  At the time of invocation,
                        031D  1084 ;       the HIRT is already owned by the caller.
                        031D  1085 ;
                        031D  1086 ;       Also entry to this routine made be made by branching (from subroutine
                        031D  1087 ;       ONLINE_COMPLETE) to labels STEP7 and STEP11.
                        031D  1088 ;
                        031D  1089 ; Inputs:
                        031D  1090 ;
                        031D  1091 ;       R3      UCB address
                        031D  1092 ;       R5      HIRT permanent CDRP address
                        031D  1093 ;
                        031D  1094 ; Implicit Inputs:
                        031D  1095 ;
                        031D  1096 ;       CDRP$L_RSPID(R5)        user RSPID
                        031D  1097 ;       HIRT$L_SAVDCDRP         original user CDRP address
                        031D  1098 ;
                        031D  1099 ;       HIRT owned by caller which implies HIRT SUBSTACK is operative
                        031D  1100 ;
                        031D  1101 ; Outputs:
                        031D  1102 ;
                        031D  1103 ;       R0      Replacement status
                        031D  1104 ;       R1      Setting for CDRP$V_ERLIP
                        031D  1105 ;       R3      UCB address (unchanged)
                        031D  1106 ;
                        031D  1107 ;       R2, R4, R5 destroyed.
                        031D  1108 ;       All other registers preserved.
                        031D  1109 ;--
                        031D  1110
                        031D  1111
                        031D  1112 DU$REPLACE_LBN::
                        031D  1113
                        031D  1114          HIRT_SUBSAVE                        ; Save callers return point on SUBSTACK.
50    001C'CF    D0     0327  1115          MOVL    HIRT$L_SAVDCDRP,R0          ; R0 => original CDRP.
  50    1C A0    D0     032C  1116          MOVL    CDRP$L_MSG_BUF(R0),R0       ; R0 => END PACKET.
        1C A0    D0     0330  1117          MOVL    MSCP$L_FRST_BAD(R0),-       ; Indicate which LBN we are
      0018'CF            0333  1118                  HIRT$L_LBN                  ;  fixing on this unit.
000A'CF    01    B0     0336  1119          MOVW    #SS$_NORMAL,-               ; Initialize worst case I/O status.
                        033B  1120                  HIRT$W_IOWORST
                        033B  1121
                        033B  1122 ; Invalidate contents of incore scratch pages.
                        033B  1123
                        033B  1124          ASSUME  HIRT$W_PG0CNTNT+2       EQ      HIRT$W_PG1CNTNT
004C'CF    01    CE     033B  1125          MNEGL   #1,HIRT$W_PG0CNTNT       ; Invalidate pages 0 and 1.
                        0340  1126
                        0340  1127          ASSUME  HIRT$W_PG2CNTNT+2       EQ      HIRT$W_PG3CNTNT
0050'CF    01    CE     0340  1128          MNEGL   #1,HIRT$W_PG2CNTNT       ; Invalidate pages 2 and 3.
                        0345  1129
                        0345  1130
                        0345  1131          ALLOC_MSG_BUF                       ; Allocate a send credit.
      03 50    E8     0348  1132          BLBS    R0,10$                      ; Branch around if successful alloc.
```

```
                                                     ; J 14
                     04BD   31   034B  1133                BRW     REPLACE_CONNECT_FAILURE ; If we had an allocation failure branch,
                                 034E  1134 10$:
                                 034E  1135
                                 034E  1136 ; STEP 4 of replacement algorithm.
                                 034E  1137 ; We clear (zero) a sector sized buffer, then read the  current
                                 034E  1138 ;   contents  of  the bad block into the buffer.  The buffer is cleared first
                                 034E  1139 ;   for the rare case when no data can be transferred (such as a  valid  sync
                                 034E  1140 ;   pattern  is not detected).  The read is performed with error recovery and
                                 034E  1141 ;   error correction enabled.  In addition to saving the  data,  we
                                 034E  1142 ;   remember  whether  or  not  the  read  succeeded.  The saved  data  is
                                 034E  1143 ;   considered valid if the read succeeded, or invalid if it did not.
                                 034E  1144 ;
                                 034E  1145
                                 034E  1146 ; First we zero out the buffer to receive the contents of the failing block.
                                 034E  1147
                           3C    BB  034E  1148          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers.
0200 8F   00   FE AF       00    2C  0350  1149          MOVC5   #0,,#0,#512,-           ; Clear page 1 prior to read.
                     0028'DF                                    @HIRT$L_PAGEIPTR
                           3C    BA  0358  1151          POPR    #^M<R2,R3,R4,R5>        ; Restore registers.
                                 035B  1152
                                 035B  1153 ; Step 4 continued.  Prepare the CDRP with SVAPTE, BOFF, and BCNT of page 1
                                 035B  1154 ;       (receiving field for upcoming read) so as to facilitate mapping of
                                 035B  1155 ;       this region.
                                 035B  1156
                     50    01    D0  035B  1157          MOVL    #1,R0                   ; Pass page to map to subroutine.
                        0748   30  035E  1158          BSBW    MAP_PAGE                ; Map page 1.
                                 0361  1159
                                 0361  1160 ; Step 4 continued. Prepare the MSCP packet to read failing block into page 1.
                                 0361  1161
                        0716   30  0361  1162          BSBW    FILL_RCT_PACKET         ; Subroutine that fills most fields in
                                 0364  1163                                          ;  MSCP packet. Returns R2=>MSCP packet.
                           21    90  0364  1164          MOVB    #MSCP$K_OP_READ,-       ; Copy the READ opcode, field not filled
                        08 A2          0366  1165                  MSCP$B_OPCODE(R2)       ;  by above subroutine.
                     0018'CF   D0  0368  1166          MOVL    HIRT$L_LBN,-            ; And also the LBN of the bad block.
                        1C A2          036C  1167                  MSCP$L_LBN(R2)
                                 036E  1168
                                 036E  1169          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                                 0371  1170
                           05    E1  0371  1171          BBC     #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                     07 09 A2          0373  1172                  MSCP$B_FLAGS(R2),15$    ;  and branch around if not.
                      0080 8F   A8  0376  1173          BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
                     000E'CF          037A  1174                  HIRT$W_STS              ;  Have been generated.
                                 037D  1175 15$:
                                 037D  1176
                                 037D  1177          UNMAP                           ; Release mapping resources.
                        2C A5    D4  0380  1178          CLRL    CDRP$L_LBUFH_AD(R5)     ; Show no mapping resources allocated.
                     52    1C A5    D0  0383  1179          MOVL    CDRP$L_MSG_BUF(R5),R2   ; Refresh R2 after unmap.
                                 0387  1180
                                 0387  1181 ; Remember status of read of failing block so as to be able to write it later
                                 0387  1182 ;       with or without the forced error flag.
                                 0387  1183
000E'CF    04   AA  0387  1184          BICW    s^#HIRT$M_FE,HIRT$W_STS ; Initialize bit.
                                 038C  1185          IF_MSCP SUCCESS, then=20$       ; Branch if READ succeeded.
000E'CF    04   A8  0392  1186          BISW    s^#HIRT$M_FE,HIRT$W_STS ; Set bit if read failed.
                                 0397  1187 20$:
                                 0397  1188
                                 0397  1189 ; Step 5.
```

```
                     0399  1190 ;  Record the data obtained when the bad block was read  during  step  4  in
                     0399  1191 ;  sector  1  of each RCT copy.  If the data cannot be successfully recorded
                     0399  1192 ;  in the RCT, report the error to the error log and go to  step  18.   Note
                     0399  1193 ;  that  the  Multi-Write algorithm used to record the data in sector 1 uses
                     0399  1194 ;  write-compare operations to  guarantee  that  the  data  is  successfully
                     0399  1195 ;  recorded.
                     0399  1196 ;
                     0399  1197
                     0399  1198 STEP5:
                     0399  1199
                     0399  1200 ; Write contents of page 1 to sector 1 of each RCT copy.
                     0399  1201
          50   01    D0  0399  1202          MOVL    #1,R0                           ; Pass sector and page number to routine.
004E'CF    50    B0  039C  1203          MOVW    R0,HIRTSW_PG1CNTNT              ; Indicate that page 1 contains RCT sector
                     03A1  1204                                                  ;  #1.
        0565    30  03A1  1205          BSBW    WRITE_RCT_BLOCK                 ; Call internal subroutine to write.
      0B 50     E8  03A4  1206          BLBS    R0,STEP6                        ; LBS implies successful write to at
                     03A7  1207          HIR_ERROR -                             ; Signal HIR error.
                     03A7  1208                  step=5, type=WRITE
        0434    31  03AF  1209          BRW     STEP18                          ; And branch to step 18.
                     03B2  1210
                     03B2  1211 ; Step 6. Record bad block's LBN, whether or not the saved data is valid and
                     03B2  1212 ;        the fact that we are now in phase 1 of replacement in sector 0 of each
                     03B2  1213 ;        RCT copy.  This means that we read sector 0 modify it and
                     03B2  1214 ;        then rewrite the updated sector to each RCT copy.  If we cannot
                     03B2  1215 ;        read any sector 0 successfully, we go to step 18.  If we cannot
                     03B2  1216 ;        successfully write at least one sector 0, we go to step 17.
                     03B2  1217
                     03B2  1218 STEP6:
          50    7C  03B2  1219          CLRQ    R0                              ; Prepare to read sector #0 into page #0.
                     03B4  1220                                                  ;  We pass the page number in R0 and
                     03B4  1221                                                  ;  the sector number in R1.
                     03B4  1222
        060C    30  03B4  1223          BSBW    READ_RCT_BLOCK                  ; Call to read RCT block.
      0B 50     E8  03B7  1224          BLBS    R0,10$                          ; LBS implies successful read.
                     03BA  1225          HIR_ERROR -                             ; Signal HIR error.
                     03BA  1226                  step=6, type=READ
        0421    31  03C2  1227          BRW     STEP18                          ; On failure goto step 18.
                     03C5  1228 10$:
    50  0024'CF    D0  03C5  1229          MOVL    HIRT$L_PAGEOPTR,R0              ; R0 => sector 0 in memory.
        0018'CF    D0  03CA  1230          MOVL    HIRT$L_LBN,-                    ; Copy bad block's LBN to RCT sector
          0C A0         03CE  1231                  RCT$L_CBN(R0)                   ;  0 copy in memory.
                     03D0  1232
08 A0     8000 8F    A8  03D0  1233          BISW    #RCT$M_RP1,RCT$W_FLAGS(R0)      ; Set bit to signal phase 1.
                   AA    03D6  1234          BICW    #RCT$M_RP2-                     ; Clear bit to signal not phase 2
                     03D7  1235                  !RCT$M_BR-                      ;  clear bad RBN flag,
                     03D7  1236                  !RCT$M_FE,-                     ;  and also clear force error before
08 A0     6080 8F        03D7  1237                  RCT$W_FLAGS(R0)                 ;  testing for valid data.
                     03DC  1238
   000E'CF    02    E1  03DC  1239          BBC     s^#HIRT$V_FE, -                 ; See if original data is valid.
                   06    03E1  1240                  HIRT$W_STS, 20$
08 A0     0080 8F    A8  03E2  1241          BISW    #RCT$M_FE, -                    ; Set force error if appropriate.
                     03E8  1242                  RCT$W_FLAGS(R0)
                     03E8  1243 20$:
          50    D4  03E8  1244          CLRL    R0                              ; Rewrite page 0.
        051C    30  03EA  1245          BSBW    WRITE_RCT_BLOCK                 ; Go to rewrite sector 0.
      0B 50     E8  03ED  1246          BLBS    R0,STEP7                        ; LBS implies successful rewrite.
```

```
                     03F0  1247          HIR_ERROR -                             ; Signal HIR error.
                     03F0  1248                  step=6, type=WRITE
                03CA 31   03F8  1249          BRW     STEP17                      ; And go to step 17.
                     03FB  1250 ; STEP7.
                     03FB  1251 ;  Write and read test patterns on the  suspected  bad  block  to  determine
                     03FB  1252 ;  whether  or  not  it  is  in  fact  a  bad  block.
                     03FB  1253 ;  Go  to  step  9  if  the  test  patterns  fail,
                     03FB  1254 ;  indicating  that  the  block  is indeed bad.  Continue with step 8 if the
                     03FB  1255 ;  test patterns succeed, indicating that the block may be good.   The  test
                     03FB  1256 ;  patterns  fail if either the block is again reported as a bad block or if
                     03FB  1257 ;  the test patterns cannot be written and read back correctly.
                     03FB  1258 ;
                     03FB  1259
                     03FB  1260 STEP7:
     52  002C'CF    D0  03FB  1261          MOVL    HIRT$L_PAGE2PTR,R2              ; R2 => target page.
 51  B6DBCB6D 8F    D0  0400  1262          MOVL    #TEST_PATTERN,R1                ; Get test pattern to write to bad block.
     50    80 8F    9A  0407  1263          MOVZBL  #512/4,R0                       ; Loop counter set to # longwords in block.
                     040B  1264 10$:
          82   51    D0  040B  1265          MOVL    R1,(R2)+                        ; Copy test pattern to page 2.
            FA 50    F5  040E  1266          SOBGTR  R0,10$                          ; Loop thru page.
                     0411  1267
          50   02    D0  0411  1268          MOVL    #2,R0                           ; Pass page to map to subroutine.
               0694  30  0414  1269          BSBW    MAP_PAGE                        ; Call to map page 2.
                     0417  1270
                     0417  1271 ; Call subroutine that recycles current END PACKET, recycles current RSPID and
                     0417  1272 ;       fills in most relevant data in the MSCP packet.
                     0417  1273
                0638 30   0417  1274          BSBW    BUILD_RCT_PACKET                ; Build a packet to transfer mapped
                     041A  1275                                                  ;  page to random LBN.
                     041A  1276
                     041A  1277          ASSUME  MSCP$W_MODIFIER  EQ  MSCP$B_OPCODE+2
08 A2    43000022 8F    D0  041A  1278          MOVL    #MSCP$K_OP_WRITE-               ; Fill in field not prepared by
                     0422  1279                  !<<MSCP$M_MD_COMP -             ;  BUILD_RCT_PACKET.
                     0422  1280                    !MSCP$M_MD_SECOR -
                     0422  1281                    !MSCP$M_MD_SEREC> @ 16>, -
                     0422  1282                  MSCP$B_OPCODE(R2)
                     0422  1283
          0018'CF    D0  0422  1284          MOVL    HIRT$L_LBN,-                    ; Fill in field filled in incorrectly
            1C A2        0426  1285                  MSCP$L_LBN(R2)                  ;  by BUILD_RCT_PACKET.
                     0428  1286          SEND_MSCP_MSG                           ; Send message to the MSCP server.
                     042B  1287
             05      E1  042B  1288          BBC     #MSCP$V_EF_ERLOG,-              ; Test for error log message generated
       07 09 A2          042D  1289                  MSCP$B_FLAGS(R2),15$            ;  and branch around if not.
        0080 8F    A8  0430  1290          BISW    #HIRT$M_ERLOGIP,-               ; Else remember that error log messages
        000E'CF        0434  1291                  HIRT$W_STS                      ;  Have been generated.
                     0437  1292 15$:
                     0437  1293          UNMAP                                   ; If write no good, give up resources.
          2C A5      D4  043A  1294          CLRL    CDRP$L_LBUFH_AD(R5)             ; And show that deallocation was done.
     52   1C A5      D0  043D  1295          MOVL    CDRP$L_MSG_BOF(R5),R2           ; Refresh R2 => END PACKET after unmap.
                     0441  1296          IF_MSCP SUCCESS, then=30$               ; Branch if WRITE successful.
                     0447  1297 20$:
               0084  31  0447  1298          BRW     STEP9                           ; Proceed to next step of replacement.
                     044A  1299 30$:
             07      E0  044A  1300          BBS     #MSCP$V_EF_BBLKR,-              ; If bad block reported again on write,
       F8 09 A2          044C  1301                  MSCP$B_FLAGS(R2),20$            ;  then branch back to proceed with
                     044F  1302                                                  ;  replacement.
                     044F  1303 40$:
```

```
                                                  M 14
                                044F  1304
                                044F  1305 ; Clear page to receive test pattern written block.
                                044F  1306
                          3C    BB  044F  1307          PUSHR   #^M<R2,R3,R4,R5>        ; Save registers.
0200 8F   00   FE AF      00    2C  0451  1308          MOVC5   #0,,#0,#512,-           ; Clear page 2 prior to read.
                        002C'DF       0459  1309                  @HIRT$L_PAGE2PTR
                          3C    BA  045C  1310          POPR    #^M<R2,R3,R4,R5>        ; Restore registers.
                                045E  1311
                                045E  1312 ; Call subroutine that recycles current END PACKET, recycles current RSPID and
                                045E  1313 ;       fills in most relevant data in the MSCP packet.
                                045E  1314
                       50   02  DO  045E  1315          MOVL    #2,R0                   ; Pass page to map to subroutine.
                          0647  30  0461  1316          BSBW    MAP_PAGE                ; Call to map page 2.
                          05EE  30  0464  1317          BSBW    BUILD_RCT_PACKET        ; Build MSCP packet to transfer page 2
                                0467  1318
                                0467  1319          ASSUME  MSCP$W_MODIFIER  EQ  MSCP$B_OPCODE+2
      08 A2   43000021 8F   DO  0467  1320          MOVL    #MSCP$K_OP_READ-        ; Fill in field not prepared by
                                046F  1321                  !<<MSCP$M_MD_COMP -     ;  BUILD_RCT_PACKET.
                                046F  1322                    !MSCP$M_MD_SECOR -
                                046F  1323                    !MSCP$M_MD_SEREC> @ 16>, -
                                046F  1324                  MSCP$B_OPCODE(R2)
                                046F  1325
                    0018'CF   DO  046F  1326          MOVL    HIRT$L_LBN,-            ; Fill in field filled in incorrectly
                      1C A2       0473  1327                  MSCP$L_LBN(R2)          ;  by BUILD_RCT_PACKET.
                                0475  1328          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                                0478  1329
                          05    E1  0478  1330          BBC     #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                   07 09 A2           047A  1331                  MSCP$B_FLAGS(R2),45$    ;  and branch around if not.
                    0080 8F   A8  047D  1332          BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
                    000E'CF       0481  1333                  HIRT$W_STS              ;  Have been generated.
                                0484  1334 45$:
                                0484  1335          UNMAP                           ; Give up MAP resources.
                       2C A5    D4  0487  1336          CLRL    CDRP$L_LBUFH_AD(R5)     ; And show that deallocation was done.
                    52 1C A5    DO  048A  1337          MOVL    CDRP$L_MSG_BOF(R5),R2   ; Refresh R2 => END PACKET after unmap.
                                048E  1338
                                048E  1339          IF_MSCP SUCCESS, then=60$       ; Branch if WRITE successful.
                                0494  1340 50$:
                          0067  31  0494  1341          BRW     STEP9                   ; On any error, goto step 9.
                                0497  1342 60$:
                          07    E0  0497  1343          BBS     #MSCP$V_EF_BBLKR,-      ; If bad block reported on read,
                   F8 09 A2           0499  1344                  MSCP$B_FLAGS(R2),50$    ;  then branch back to proceed with
                                049C  1345                                          ;  replacement of same.
                52   002C'CF   DO  049C  1346          MOVL    HIRT$L_PAGE2PTR,R2      ; R2 => target page.
            51  B6DBCB6D 8F   DO  04A1  1347          MOVL    #TEST_PATTERN,R1        ; Test pattern to compare to bad block.
                50   80 8F    9A  04A8  1348          MOVZBL  #512/4,R0               ; Loop counter set to # longwords in block.
                                04AC  1349 70$:
                       82   51  D1  04AC  1350          CMPL    R1,(R2)+                ; Compare test pattern to page 2.
                          E3    12  04AF  1351          BNEQ    50$                     ; On any discrepancy, branch.
                          F8 50 F5  04B1  1352          SOBGTR  R0,70$                  ; Loop thru page.
                                04B4  1353
                                04B4  1354 ; STEP8.
                                04B4  1355 ;  We write the saved data back out  to  the  bad  block  using  a
                                04B4  1356 ;  write-compare  operation.  The write-compare is performed with the "force
                                04B4  1357 ;  error" modifier if and only if the saved data is invalid.  Go to step  13
                                04B4  1358 ;  if the write-compare both succeeds AND the block is no longer reported as
                                04B4  1359 ;  a bad block -- the original problem was a transient.   The  write-compare
                                04B4  1360 ;  succeeds if no error is detected and the saved data is valid or if only a
```

```
                                04B4  1361 ;  forced error is detected and the saved data is invalid.
                                04B4  1362 ;
                                04B4  1363
                                04B4  1364 STEP8:
                                04B4  1365
                                04B4  1366 ; Try to write original data out to block originally reported as bad since
                                04B4  1367 ;       error now appears to have been transient.
                                04B4  1368
                 50   01   D0  04B4  1369          MOVL    #1,R0                   ; Data from bad block is in page 1.
                    05F1   30  04B7  1370          BSBW    MAP_PAGE                ; Map page 1.
                    0598   30  04BA  1371          BSBW    BUILD_RCT_PACKET        ; Recycle etc., and fill in packet.
                               04BD  1372
                               04BD  1373          ASSUME  MSCP$W_MODIFIER  EQ  MSCP$B_OPCODE+2
                           D0  04BD  1374          MOVL    #MSCP$K_OP_WRITE-       ; Fill in field not prepared by
                               04BE  1375                  !<MSCP$M_MD_COMP@16>,-  ;  BUILD_RCT_PACKET.
          08 A2   40000022 8F  04BE  1376                  MSCP$B_OPCODE(R2)
                               04C5  1377
              000E'CF   02     E1  04C5  1378          BBC     S^#HIRT$V_FE, -         ; See if original data is valid.
                        06         04CA  1379                  HIRT$W_STS, 10$
             0A A2    1000 8F     A8  04CB  1380          BISW    #MSCP$M_MD_ERROR, -     ; Set force error modifier if
                               04D1  1381                  MSCP$W_MODIFIER(R2)     ;  original data is invalid.
                               04D1  1382 10$:
                  0018'CF   D0  04D1  1383          MOVL    HIRT$L_LBN,-            ; Fill in field filled in incorrectly
                     1C A2      04D5  1384                  MSCP$L_LBN(R2)          ;  by BUILD_RCT_PACKET.
                               04D7  1385          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                               04DA  1386
                        05     E1  04DA  1387          BBC     #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                  07 09 A2         04DC  1388                  MSCP$B_FLAGS(R2),15$    ;  and branch around if not.
                   0080 8F     A8  04DF  1389          BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
                   000E'CF         04E3  1390                  HIRT$W_STS              ;  Have been generated.
                               04E6  1391 15$:
                               04E6  1392          UNMAP                           ; If write no good, give up resources.
                      2C A5    D4  04E9  1393          CLRL    CDRP$L_LBUFH_AD(R5)     ; And show that deallocation was done.
                52    1C A5    D0  04EC  1394          MOVL    CDRP$L_MSG_BOF(R5),R2   ; Refresh R2 => END PACKET after unmap.
                               04F0  1395
                               04F0  1396          IF_MSCP FAILURE, then=STEP9     ; Branch if problem not transient.
                        07     E0  04F6  1397          BBS     #MSCP$V_EF_BBLKR,-      ; If bad block reported on write,
                  03 09 A2         04F8  1398                  MSCP$B_FLAGS(R2),STEP9  ;  then branch ahead to proceed with
                               04FB  1399                                          ;  replacement of same.
                    0182   31     04FB  1400          BRW     STEP13                  ; Branch if error was transient.
                               04FE  1401
                               04FE  1402 ; STEP9.
                               04FE  1403 ;  We scan the RCT and determine  what  new  RBN  the  bad  block
                               04FE  1404 ;  should  be  replaced  with,  whether  or  not  the the bad block has been
                               04FE  1405 ;  previously replaced, and (if it has previously  been  replaced)  the  bad
                               04FE  1406 ;  block's  old  RBN.  The RCT is NOT updated at this time.  If the RCT scan
                               04FE  1407 ;  fails, we report the error to the error log and go to step 16.
                               04FE  1408 ;
                               04FE  1409
                               04FE  1410
                               04FE  1411 STEP9:
                        05DA   30  04FE  1412          BSBW    SEARCH_RCT              ; Routine to search the RCT for an RBN.
                     22 50     E8  0501  1413          BLBS    R0,STEP10               ; LBC implies success, so goto step 10.
              000E'CF    06     E0  0504  1414          BBS     S^#HIRT$V_RCTFULL, -    ; Check for RCTFULL error and
                         0A         0509  1415                  HIRT$W_STS, 910$        ; branch if that is the problem.
                               050A  1416          HIR_ERROR -                     ; Else, signal HIR READ error.
                               050A  1417                  step=9, type=READ
```

```
                     0F   11  0512  1418                BRB     980$                    ; Join common branch to step 16.
                                0514  1419 910$:        HIR_ERROR -
                                0514  1420                       step=9, type=RCTFULL    ; Signal HIR RCTFULL error.
000A'CF   216C 8F    B0      051C  1421                MOVW    #SS$_BADRCT, -          ; Also, supply a worst case error
                                0523  1422                       HIRT$W_IOWORST          ; status.
                   025D   31  0523  1423 980$:        BRW     STEP16                  ; Go to step 16 after any failure.
                                0526  1424
                                0526  1425 ; STEP10.
                                0526  1426 ;  Record the new RBN, whether or not the  bad  block  has  been  previously
                                0526  1427 ;  replaced,  the  bad block's old RBN (if it has been previously replaced),
                                0526  1428 ;  and the fact that we are in phase 2 of bad block replacement in sector  0
                                0526  1429 ;  of  each  RCT  copy.   The  RCT must be updated without reading sector 0,
                                0526  1430 ;  instead using the copy of sector 0 last read from or written to the  RCT.
                                0526  1431 ;  If the RCT cannot be updated, report the error to the error log and go to
                                0526  1432 ;  step 16.
                                0526  1433 ;
                                0526  1434
                                0526  1435
                                0526  1436 STEP10:
          50  0024'CF   D0      0526  1437                MOVL    HIRT$L_PAGE0PTR,R0      ; R0 => Page 0.
              0058'CF   D0      052B  1438                MOVL    HIRT$L_RBN,-            ; Update date to sector 0 copy in
                10 A0           052F  1439                        RCT$L_RBN(R0)           ;  in memory.
000E'CF         03   E1         0531  1440                BBC     s^#HIRT$V_MATCH, -      ; See if we had a failing RBN, and
                   0C           0536  1441                        HIRT$W_STS,10$          ;  if NOT, branch around.
              2000 8F   A8      0537  1442                BISW    #RCT$M_BR,-             ; Indicate failing RBN in sector 0
                08 A0           053B  1443                        RCT$W_FLAGS(R0)         ;  flags word.
              005C'CF   D0      053D  1444                MOVL    HIRT$L_MATCHRBN,-       ; And also indicate the failing RBN.
                14 A0           0541  1445                        RCT$L_BAD_RBN(R0)
                                0543  1446 10$:
              8000 8F   AA      0543  1447                BICW    #RCT$M_RP1,-            ; Show that we are leaving phase 1
                08 A0           0547  1448                        RCT$W_FLAGS(R0)         ;  of replacement processing.
              4000 8F   A8      0549  1449                BISW    #RCT$M_RP2,-            ; And entering phase 2.
                08 A0           054D  1450                        RCT$W_FLAGS(R0)
                                054F  1451
                   50   D4      054F  1452                CLRL    R0                      ; Rewrite page 0.
                 03B5   30      0551  1453                BSBW    WRITE_RCT_BLOCK         ; Go write the sector.
                0B 50   E8      0554  1454                BLBS    R0,STEP11               ; If success, go to next step.
                                0557  1455                HIR_ERROR -                     ; Signal HIR error.
                                0557  1456                        step=10, type=WRITE
                   0221   31      055F  1457                BRW     STEP16                  ; Branch on failure.
                                0562  1458
                                0562  1459 ; STEP11.
                                0562  1460 ;  We update the RCT to indicate  that  the  bad  block  has  been
                                0562  1461 ;  replaced with the new RBN, and that the old RBN (if any) is unusable.  If
                                0562  1462 ;  this requires updating two blocks in the RCT, then both  blocks  must  be
                                0562  1463 ;  read  before  either is written.  If a block cannot be read successfully,
                                0562  1464 ;  report the error to the error log and go to step 16.  If a  block  cannot
                                0562  1465 ;  be written successfully, report the error to the error log and go to step
                                0562  1466 ;  15.
                                0562  1467 ;
                                0562  1468
                                0562  1469
                                0562  1470 STEP11:
50  0058'CF   07   00   EF      0562  1471                EXTZV   #0,#7,HIRT$L_RBN,R0     ; R0 = offset in sector of RBN descriptor.
          51  002C'CF   D0      0569  1472                MOVL    HIRT$L_PAGE2PTR,R1      ; R1 => sector containing RBN descriptor.
              50   6140   DE      056E  1473                MOVAL   (R1)[R0],R0             ; R0 => RBN descriptor.
                                0572  1474
```

```
                20000000 8F    C9  0572  1475          BISL3   #RCT$M_ALLOCATED,-       ; Put LBN being replaced into descriptor.
              60   0018'CF                0578  1476                  HIRT$L_LBN(R0)           ;  and or in ALLOCATED bit.
              000E'CF    04    E1  057C  1477          BBC     s^#HIRT$V_EMPTYPE, -     ; Branch if primary RBN allocation.
                         07          0581  1478                  HIRT$W_STS,10$
           60   10000000 8F    C8  0582  1479          BISL    #RCT$M_NONPRIME,(R0)     ; Indicate non prime allocation.
                                    0589  1480 10$:
              000E'CF    03    E0  0589  1481          BBS     s^#HIRT$V_MATCH, -       ; Branch if RCT search showed RBN failed.
                         03          058E  1482                  HIRT$W_STS,20$
                       0063  31    058F  1483          BRW     60$                      ; If NOT RBN failure, skip RBN
                                    0592  1484                                           ;  descriptor update.
                                    0592  1485 20$:
        51    005C'CF    F9 8F    78  0592  1486          ASHL    #-7,HIRT$L_MATCHRBN,R1   ; R1 = relative RCT block containing
                                    0599  1487                                           ;  bad RBN descriptor.
                    51    02    C0  0599  1488          ADDL    #2,R1                    ; Add in sectors 0 and 1.
              0050'CF    51    B1  059C  1489          CMPW    R1,HIRT$W_PG2CNTNT       ; Page 2 contains RBN descriptor of
                                    05A1  1490                                           ;  allocatable RBN and maybe also
                                    05A1  1491                                           ;  descriptor of bad RBN.
                          1B    13  05A1  1492          BEQL    40$                      ; EQL implies both descriptors in same
                                    05A3  1493                                           ;  block.
                       50   03    D0  05A3  1494          MOVL    #3,R0                    ; Indicate that we want to read into
                                    05A6  1495                                           ;  page 3.
                         041A  30    05A6  1496          BSBW    READ_RCT_BLOCK           ; Read sector (R1) into page 3.
                        0B 50  E8    05A9  1497          BLBS    R0,30$                   ; If success, continue.
                                    05AC  1498          HIR_ERROR -                      ; Signal HIR error.
                                    05AC  1499                  step=11, type=READ
                         01CC  31    05B4  1500          BRW     STEP16                   ; Branch on failure.
                                    05B7  1501
                                    05B7  1502 30$:
              52   0030'CF    D0    05B7  1503          MOVL    HIRT$L_PAGE3PTR,R2       ; R2 => page with bad RBN descriptor.
                         05    11    05BC  1504          BRB     50$                      ; Branch around.
                                    05BE  1505 40$:
              52   002C'CF    D0    05BE  1506          MOVL    HIRT$L_PAGE2PTR,R2       ; R2 => page with bad RBN descriptor.
                                    05C3  1507 50$:
        50    005C'CF    07    00    EF  05C3  1508          EXTZV   #0,#7,HIRT$L_MATCHRBN,R0; R0 = offset of bad RBN descriptor.
                    50    6240  DE    05CA  1509          MOVAL   (R2)[R0],R0              ; R0 => bad RBN descriptor.
              0060'CF    60    D0    05CE  1510          MOVL    (R0),HIRT$L_BADRBND      ; Save Bad RBN descriptor in case
                                    05D3  1511                                           ;  we have to restore due to failure.
           60   40000000 8F    D0    05D3  1512          MOVL    #RCT$M_UNUSABLE,(R0)     ; Clear LBN and mark unusable bit in
                                    05DA  1513                                           ;  descriptor.
              002C'CF    52    D1    05DA  1514          CMPL    R2,HIRT$L_PAGE2PTR       ; See if both descriptors in same page.
                         14    13    05DF  1515          BEQL    60$                      ; EQL implies yes. Go do only 1 write.
                                    05E1  1516
                       50   03    D0    05E1  1517          MOVL    #3,R0                    ; Rewrite page 3 [R0].
                         0322  30    05E4  1518          BSBW    WRITE_RCT_BLOCK          ; Go write.
                        0B 50  E8    05E7  1519          BLBS    R0,60$                   ; If success, continue.
                                    05EA  1520          HIR_ERROR -                      ; Signal HIR error.
                                    05EA  1521                  step=11, type=WRITE
                         00FA  31    05F2  1522          BRW     STEP15_A                 ; Branch on failure.
                                    05F5  1523
                                    05F5  1524 60$:
                       50   02    D0    05F5  1525          MOVL    #2,R0                    ; Prepare to write page 2.
                         030E  30    05F8  1526          BSBW    WRITE_RCT_BLOCK          ; Go write.
                        0B 50  E8    05FB  1527          BLBS    R0,STEP12                ; If success, goto next step.
                                    05FE  1528          HIR_ERROR -                      ; Signal HIR error.
                                    05FE  1529                  step=11, type=WRITE
                         010E  31    0606  1530          BRW     STEP15_B                 ; Branch on failure.
                                    0609  1531
```

```
                                0609  1532 ; STEP12.
                                0609  1533 ;  We use the REPLACE command to revector the  bad  block  to  the
                                0609  1534 ;  chosen replacement block, then use the standard WRITE command (addressed
                                0609  1535 ;  to the bad block's LBN) with the "compare" modifier asserted to store the
                                0609  1536 ;  saved data in the replacement block.  The write-compare is performed with
                                0609  1537 ;  the "force error" modifier if and only if  the  saved  data  is  invalid.
                                0609  1538 ;  Note  that  the  REPLACE command implicitly verifies that a head or servo
                                0609  1539 ;  track failure has not  occurred,  causing  a  large  number  of  improper
                                0609  1540 ;  replacements.  If the REPLACE command fails, go to step 15.  If the WRITE
                                0609  1541 ;  command fails, go to step 9 to re-scan the RCT  for  another  RBN.   Note
                                0609  1542 ;  that  the  current  new  RBN  will become the old RBN for this next pass.
                                0609  1543 ;  Either failure will have already been reported to  the  error  log.   The
                                0609  1544 ;  WRITE  command  succeeds  if  no  error is detected and the saved data is
                                0609  1545 ;  valid or if only a forced  error  is  detected  and  the  saved  data  is
                                0609  1546 ;  invalid.
                                0609  1547 ;
                                0609  1548
                                0609  1549 STEP12:
                                0609  1550         RECYCH_MSG_BUF                  ; Recycle END PACKET into MSCP buffer.
                     03 50   E8 060C  1551         BLBS    R0,5$                   ; LBS means allocation success.
                      01F9   31 060F  1552         BRW     REPLACE_CONNECT_FAILURE ; Allocation failure means CONNECTION
                                0612  1553                                         ;  failure.
                                0612  1554 5$:
                                0612  1555         INIT_MSCP_MSG ucb=(R3)          ; Initialize MSCP packet for REPLACE.
                                0615  1556
                                0615  1557         ASSUME  MSCP$W_MODIFIER  EQ  MSCP$B_OPCODE+2
                           D0   0615  1558         MOVL    #MSCP$K_OP_REPLC-       ; Fill in field not prepared by
                                0616  1559                 !<MSCP$M_MD_EXPRS@16>,- ;  BUILD_RCT_PACKET.
           08 A2    80000014 8F  0616  1560                 MSCP$B_OPCODE(R2)
                                0611  1561
                 000E'CF    04   E0 061D  1562         BBS     s^#HIRT$V_EMPTYPE, -    ; See if primary or secondary RBN,
                            04        0622  1563                 HIRT$W_STS,10$          ;  branch if secondary.
                            01   A8 0623  1564         BISW    #MSCP$M_MD_PRIMR,-      ; Set primary modifier if
                         0A A2        0625  1565                 MSCP$W_MODIFIER(R2)     ;  called for.
                                0627  1566 10$:
              0C A2    0058'CF   D0 0627  1567         MOVL    HIRT$L_RBN,MSCP$L_RBN(R2); Fill in special REPLACE field.
                       0018'CF   D0 062D  1568         MOVL    HIRT$L_LBN,-            ; Fill in field filled in incorrectly
                          1C A2        0631  1569                 MSCP$L_LBN(R2)          ;  by BUILD_RCT_PACKET.
                                0633  1570         SEND_MSCP_MSG                   ; Send message to the MSCP server.
                                0636  1571
                            05   E1 0636  1572         BBC     #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                     07 09 A2        0638  1573                 MSCP$B_FLAGS(R2),15$    ;  and branch around if not.
                       0080 8F   A8 063B  1574         BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
                       000E'CF        063F  1575                 HIRT$W_STS              ;  Have been generated.
                                0642  1576 15$:
                                0642  1577         IF_MSCP_SUCCESS, then=20$       ; Branch if REPLACE was successful.
                                0648  1578         HIR_ERROR -                     ; Signal HIR error.
                                0648  1579                 step=12, type=REPFAIL
                          00C4  31 0650  1580         BRW     STEP15_B
                                0653  1581 20$:
                     50   01   D0 0653  1582         MOVL    #1,R0                   ; Data from bad block is in page 1.
                        0452  30 0656  1583         BSBW    MAP_PAGE                ; Map page 1.
                        03F9  30 0659  1584         BSBW    BUILD_RCT_PACKET        ; Recycle etc., and fill in packet.
                                065C  1585
                                065C  1586         ASSUME  MSCP$W_MODIFIER  EQ  MSCP$B_OPCODE+2
                           D0   065C  1587         MOVL    #MSCP$K_OP_WRITE-       ; Fill in field not prepared by
                                065D  1588                 !<MSCP$M_MD_COMP@16>,-  ;  BUILD_RCT_PACKET.
```

```
                08 A2   40000022 8F        0065D  1589                    MSCP$B_OPCODE(R2)
                                            0664  1590
                    000E'CF   02   E1       0664  1591            BBC     s^#HIRT$V_FE,-          ; See if original data is valid.
                              06            0669  1592                    HIRT$W_STS,30$
                 0A A2    1000 8F   A8      066A  1593            BISW    #MSCP$M_MD_ERROR,-      ; Set force error modifier if
                                            0670  1594                    MSCP$W_MODIFIER(R2)     ;  original data is invalid.
                                            0670  1595 30$:
                        0018'CF   D0        0670  1596            MOVL    HIRT$L_LBN,-            ; Fill in field filled in incorrectly
                             1C A2          0674  1597                    MSCP$L_LBN(R2)          ;  by BUILD_RCT_PACKET.
                                            0676  1598            SEND_MSCP_MSG                   ; Send message to the MSCP server.
                                            0679  1599
                             05   E1        0679  1600            BBC     #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                          07 09 A2          067B  1601                    MSCP$B_FLAGS(R2),35$    ;  and branch around if not.
                           0080 8F   A8     067E  1602            BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
                           000E'CF          0682  1603                    HIRT$W_STS              ;  Have been generated.
                                            0685  1604 35$:
                                            0685  1605            UNMAP                           ; If write no good, give up resources.
                             2C A5   D4     0688  1606            CLRL    CDRP$L_LBUFH_AD(R5)     ; And show that deallocation was done.
                        52   1C A5   D0     068B  1607            MOVL    CDRP$L_MSG_BUF(R5),R2   ; Refresh R2 => END PACKET after unmap.
                                            068F  1608
                                            068F  1609            IF_MSCP SUCCESS, then=STEP13    ; If WRITE successful go to step 13.
                                            0695  1610
                                            0695  1611 ;          ASSUME that the force data subcode is zero
                             08   B1        0695  1612            CMPW    #MSCP$K_ST_DATA,-       ; See if data error with force error
                          0A A2             0697  1613                    MSCP$W_STATUS(R2)       ;  subcode.
                             0A   12        0699  1614            BNEQ    50$                     ; If NOT, then branch to take action.
                                            069B  1615
                    000E'CF   02   E0       0698  1616            BBS     s^#HIRT$V_FE,-          ; To STEP13 if force error expected.
                              0F            06A0  1617                    HIRT$W_STS,STEP13
                                            06A1  1618            BUG_CHECK       DISKCLASS,FATAL         ; Shouldn't happen.
                                            06A5  1619 50$:
                                            06A5  1620            HIR_ERROR -                     ; Signal HIR error.
                                            06A5  1621                    step=12, type=WRITE
                         FE4E   31          06AD  1622            BRW     STEP9                   ; Following algorithm, goto step 9.
                                            06B0  1623
                                            06B0  1624 ; STEP13.
                                            06B0  1625 ;  We update sector 0 of the RCT copies to indicate that we are no
                                            06B0  1626 ;  longer  in  the middle of replacing a bad block.  The RCT must be updated
                                            06B0  1627 ;  without reading sector 0, instead using the copy of sector  0  last  read
                                            06B0  1628 ;  from  or  written  to  the RCT.  If the RCT cannot be updated, report the
                                            06B0  1629 ;  error to the error log and go to step 17.
                                            06B0  1630 ;
                                            06B0  1631
                                            06B0  1632 STEP13:
                        50   0024'CF   D0     06B0  1633            MOVL    HIRT$L_PAGE0PTR,R0      ; R0 => page 0, which contains sector 0.
                                  AA        06B5  1634            BICW    #RCT$M_RP1-             ; Reset flags in sector zero.  We are
                                            06B6  1635                    !RCT$M_RP2-             ;  NOT in phase 1 nor in phase 2.
                                            06B6  1636                    !RCT$M_BR-              ;  Also we clear Bad RBN flag and
                                            06B6  1637                    !RCT$M_FE,-             ;   force error flag as well.
                08 A0   E080 8F             06B6  1638                    RCT$W_FLAGS(R0)
                                            06B8  1639
                                            06B8  1640 ;          ASSUME  RCT$L_RBN       EQ      RCT$L_LBN+4
                             0C A0   7C     06B8  1641            CLRQ    RCT$L_LBN(R0)           ; Zero out RBN and LBN.
                             14 A0   D4     06BE  1642            CLRL    RCT$L_BAD_RBN(R0)       ; Also clear BADRBN field.
                                            06C1  1643
                               50   D4     06C1  1644            CLRL    R0                      ; Rewrite page 0,
                            0243    30      06C3  1645            BSBW    WRITE_RCT_BLOCK         ; Go write page into sector.
```

```
                08 50    E8  06C6  1646          BLBS    R0,STEP14               ; LBS is success.
                             06C9  1647          HIR_ERROR -                     ; Signal HIR error.
                             06C9  1648                  step=13, type=WRITE
                 00F1    31  06D1  1649          BRW     STEP17                  ; After failure goto step 17.
                             06D4  1650
                             06D4  1651 ; STEP14.
                             06D4  1652 ;  We set the success return code and return to our internal caller.
                             06D4  1653 ;
                             06D4  1654
                             06D4  1655 STEP14:
                             06D4  1656          DEALLOC_MSG_BUF
                       51    D4  06D7  1657          CLRL    R1                      ; Prepare to return ERLIP bit if set.
                       07    E5  06D9  1658          BBCC    s^#HIRT$V_ERLOGIP,-     ; Branch around if clear and clear if
          03 000E'CF           06DB  1659                  HIRT$W_ST5,10$          ;  already set.
                 51    04    A8  06DF  1660          BISW    #CDRP$M_ERLIP,R1        ; Set bit in R1 so as to return to caller.
                             06E2  1661 10$:
                 50    01    3C  06E2  1662          MOVZWL  S^#SS$_NORMAL,R0        ; Prepare to return status to caller.
                             06E5  1663          HIRT_SUBRETURN                  ; Return to caller.
                             06EF  1664
                             06EF  1665 ; STEP15 - If here we failed to update the RCT (STEP11) or we failed in the
                             06EF  1666 ;          REPLACE (STEP12).
                             06EF  1667 ;  We restore the RCT to indicate that the new RBN is  unallocated
                             06EF  1668 ;  and usable and that the bad block is either not replaced or revectored to
                             06EF  1669 ;  the old RBN, whichever was it's original status.  The RCT must be updated
                             06EF  1670 ;  without  reading  any  blocks  from  it,  instead using the copies of the
                             06EF  1671 ;  relevent blocks that were read from the RCT in step 11.  Any  errors  are
                             06EF  1672 ;  reported to the error log but otherwise ignored.
                             06EF  1673 ;
                             06EF  1674
                             06EF  1675 STEP15_A:
                             06EF  1676
                             06EF  1677 ; Here we failed to write the RCT block containing the Bad RBN only when
                             06EF  1678 ;       this descriptor resided in a different RCT block than that of the
                             06EF  1679 ;       selected RBN.   Because of the way that STEP11 works the copy
                             06EF  1680 ;       of the RCT Block is in page 3.
                             06EF  1681
                   52  0030'CF    D0  06EF  1682          MOVL    HIRT$L_PAGE3PTR,R2      ; R2 => copy of page 3 in memory.
 50    005C'CF    07    00    EF  06F4  1683          EXTZV   #0,#7,HIRT$L_MATCHRBN,R0; R0 = offset (longword) of Bad RBN
                             06FB  1684                                          ;  descriptor in page 3.
                 50  6240    DE  06FB  1685          MOVAL   (R2)[R0],R0             ; R0 => Bad RBN descriptor slot.
               60  0060'CF    D0  06FF  1686          MOVL    HIRT$L_BADRBND,(R0)     ; Restore Bad RBN descriptor.
                   50    03    D0  0704  1687          MOVL    #3,R0                   ; Prepare to try to rewrite page 3.
                      01FF    30  0707  1688          BSBW    WRITE_RCT_BLOCK         ; Try to rewrite.
                   08 50    E8  070A  1689          BLBS    R0, 90$                 ; Branch if WRITE succeeded.
                             070D  1690          HIR_ERROR -                     ; Else, signal HIR error.
                             070D  1691                  step=15, type=WRITE
                      6C    11  0715  1692 90$:     BRB     STEP16                  ; Always go to step 16.
                             0717  1693
                             0717  1694 STEP15_B:
                             0717  1695
                             0717  1696 ; If here we failed in the REPLACE operation or in updating the RCT block
                             0717  1697 ;       containing the selected RBN descriptor.  So we try to restore the
                             0717  1698 ;       RCT sector(s) that contained the RBN descriptor(s).  If there was
                             0717  1699 ;       no Bad RBN, then we simply want to restore the contents of page 2
                             0717  1700 ;       to the sector indicated by HIRT$W_PG2CNTNT, after clearing the
                             0717  1701 ;       RBN descriptor slot associated with the selected RBN.
                             0717  1702
```

```
                000E'CF    03   E1  0717  1703          BBC     s^#HIRT$V_MATCH, -      ; If NO Bad RBN, branch ahead to
                           43      071C  1704                  HIRT$W_STS,20$          ;  restore only one descriptor.
                                   071D  1705
                                   071D  1706 ; If there was a Bad RBN, and its descriptor happened to reside in the
                                   071D  1707 ;       same RCT sector as the descriptor of the selected RBN, then we first
                                   071D  1708 ;       first restore the old contents of the Bad RBN descriptor then
                                   071D  1709 ;       if this descriptor resided in a different sector than the selected
                                   071D  1710 ;       RBN's descriptor, we rewrite this other sector first.  Note the
                                   071D  1711 ;       other sector is contained in page 3 while the selected
                                   071D  1712 ;       RBN's descriptor is always in page 2.
                                   071D  1713
                     52  002C'CF   D0  071D  1714          MOVL    HIRT$L_PAGE2PTR,R2      ; R2 => copy of page 2.
           51  005C'CF    F9 8F    78  0722  1715          ASHL    #-7,HIRT$L_MATCHRBN,R1  ; Calculate RCT sector # for Bad RBN.
                        51   02    C0  0729  1716          ADDL    #2,R1                   ; Add in RCT sectors 0 and 1.
                  0050'CF   51    B1  072C  1717          CMPW    R1,HIRT$W_PG2CNTNT      ; See if in same sector.
                            05    13  0731  1718          BEQL    10$                     ; EQL implies yes.
                                   0733  1719
                     52  0030'CF   D0  0733  1720          MOVL    HIRT$L_PAGE3PTR,R2      ; R2 => copy of page 3.
                                   0738  1721 10$:
           50  005C'CF   07   00   EF  0738  1722          EXTZV   #0,#7,HIRT$L_MATCHRBN,R0; R0 = offset (longword) of Bad RBN
                                   073F  1723                                          ;  descriptor in page 2 or 3.
                        50  6240   DE  073F  1724          MOVAL   (R2)[R0],R0             ; R0 => Bad RBN descriptor slot.
                  60  0060'CF   D0  0743  1725          MOVL    HIRT$L_BADRBND,(R0)     ; Restore Bad RBN descriptor.
                  002C'CF   52   D1  0748  1726          CMPL    R2,HIRT$L_PAGE2PTR      ; See if we have to do both pages.
                            11    13  074D  1727          BEQL    20$                     ; EQL implies NO, only page 2.
                        50  03   D0  074F  1728          MOVL    #3,R0                   ; Prepare to try to rewrite page 3.
                          01B4   30  0752  1729          BSBW    WRITE_RCT_BLOCK         ; Try to rewrite.
                        08 50    E8  0755  1730          BLBS    R0, 20$                 ; Branch if WRITE succeeded.
                                   0758  1731          HIR_ERROR -                     ; Else, signal HIR error.
                                   0758  1732                  step=15, type=WRITE
                                   0760  1733 20$:
                                   0760  1734
                                   0760  1735 ; Here we clear the selected RBN's descriptor and rewrite the sector from
                                   0760  1736 ;       page 2.
                                   0760  1737
                     52  002C'CF   D0  0760  1738          MOVL    HIRT$L_PAGE2PTR,R2      ; R2 => page 2 in memory.
           50  0058'CF   07   00   EF  0765  1739          EXTZV   #0,#7,HIRT$L_RBN,R0     ; R0 = offset (longword) of selected RBN
                                   076C  1740                                          ;  descriptor in page 2.
                        50  6240   DE  076C  1741          MOVAL   (R2)[R0],R0             ; R0 => RBN descriptor slot.
                            60    D4  0770  1742          CLRL    (R0)                    ; Restore to available RBN descriptor.
                                   0772  1743
                        50  02     D0  0772  1744          MOVL    #2,R0                   ; Prepare to try to rewrite page 2.
                          0191     30  0775  1745          BSBW    WRITE_RCT_BLOCK         ; Try to rewrite.
                        08 50      E8  0778  1746          BLBS    R0, STEP16              ; Branch if WRITE succeeded.
                                   077B  1747          HIR_ERROR -                     ; Else, signal HIR error.
                                   077B  1748                  step=15, type=WRITE
                                   0783  1749 ; ----- BRB     STEP16                  ; Always continue with step 16.
                                   0783  1750
                                   0783  1751 ; STEP16.
                                   0783  1752 ;  We use the standard WRITE command (addressed to the bad block's
                                   0783  1753 ;  LBN)  to  restore the saved data.  The write is performed with the "force
                                   0783  1754 ;  error" modifier if and only if the saved data is invalid.  Any errors are
                                   0783  1755 ;  reported to the error log but otherwise ignored.
                                   0783  1756 ;
                                   0783  1757
                                   0783  1758 STEP16:
                        50  01     D0  0783  1759          MOVL    #1,R0                   ; Prepare to try to write original data
```

```
                        0786                                    ;  to bad block that we could not replace.
          0322   30     0786            BSBW    MAP_PAGE                ; Map page 1.
          02C9   30     0789            BSBW    BUILD_RCT_PACKET        ; Build MSCP packet.
                        078C
                        078C            ASSUME  MSCP$W_MODIFIER EQ      MSCP$B_OPCODE+2
            22   D0     078C            MOVL    #MSCP$K_OP_WRITE,-      ; Put in write opcode.
         08 A2          078E                    MSCP$B_OPCODE(R2)
  000E'CF   02   E1     0790            BBC     s^#HIRT$V_FE, -         ; Branch around if NO forced error.
            06          0795                    HIRT$W_STS,10$
       1000 8F   A8     0796            BISW    #MSCP$M_MD_ERROR,-      ; Set forced error bit modifier on.
         0A A2          079A                    MSCP$W_MODIFIER(R2)
                        079C    10$:
       0018'CF   D0     079C            MOVL    HIRT$L_LBN,-            ; Indicate LBN to write.
         1C A2          07A0                    MSCP$L_LBN(R2)
                        07A2            SEND_MSCP_MSG                   ; Send message to the MSCP server.
                        07A5            IF_MSCP SUCCESS, then=12$       ; Branch if WRITE succeeded.
                        07AB            HIR_ERROR -                     ; Else, signal HIR error.
                        07AB                    step=16, type=WRITE
   07 09 A2     05   E1 07B3    12$:    BBC     #MSCP$V_EF_ERLOG, -     ; Test for error log message generated
                        07B8                    MSCP$B_FLAGS(R2),15$    ;  and branch around if not.
       0080 8F   A8     07B8            BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
       000E'CF          07BC                    HIRT$W_STS              ;  Have been generated.
                        07BF    15$:
                        07BF            UNMAP
         2C A5   D4     07C2            CLRL    CDRP$L_LBUFH_AD(R5)     ; Indicate no mapping resources.
                        07C5
                        07C5    ; STEP17.
                        07C5    ;  We update sector 0 of the RCT copies to indicate that it is  no
                        07C5    ;  longer  in  the middle of replacing a bad block.  The RCT must be updated
                        07C5    ;  without reading sector 0, instead using the copy of sector  0  last  read
                        07C5    ;  from or written to the RCT.  Any errors are reported to the error log but
                        07C5    ;  otherwise ignored.
                        07C5    ;
                        07C5
                        07C5
                        07C5    STEP17:
    50    0024'CF   D0  07C5            MOVL    HIRT$L_PAGEOPTR,R0      ; R0 => page 0, which contains sector 0.
                    AA  07CA            BICW    #RCT$M_RP1-             ; Reset flags in sector zero.  We are
                        07CB                    !RCT$M_RP2-             ;  NOT in phase 1 nor in phase 2.
                        07CB                    !RCT$M_BR-              ;  Also we clear Bad RBN flag and
                        07CB                    !RCT$M_FE,-             ;   force error flag as well.
 08 A0    E080 8F       07CB                    RCT$W_FLAGS(R0)
                        07D0
                        07D0    ;       ASSUME  RCT$L_RBN       EQ      RCT$L_LBN+4
         0C A0   7C     07D0            CLRQ    RCT$L_LBN(R0)           ; Zero out RBN and LBN.
         14 A0   D4     07D3            CLRL    RCT$L_BAD_RBN(R0)       ; Also clear BADRBN field.
                        07D6
            50   D4     07D6            CLRL    R0                      ; Rewrite page 0.
          012E   30     07D8            BSBW    WRITE_RCT_BLOCK         ; Go write page into sector.
         08 50   E8     07DB            BLBS    R0, STEP18              ; Branch if WRITE successful.
                        07DE            HIR_ERROR -                     ; Else, signal HIR error.
                        07DE                    step=17, type=WRITE
                        07E6
                        07E6    ; STEP18.
                        07E6    ;  We set the failure return code and return to our internal caller.
                        07E6    ;
                        07E6
```

```
                                07E6  1817 STEP18:
                                07E6  1818              DEALLOC_MSG_BUF
                          51    D4  07E9  1819              CLRL    R1                      ; Prepare to return ERLOGIP bit if set.
                000E'CF   07    E5  07EB  1820              BBCC    s^#HIRT$V_ERLOGIP, -    ; Branch around if clear and clear if
                          03        07F0  1821                      HIRT$W_STS,10$          ;  already set.
                     51   04    A8  07F1  1822              BISW    #CDRP$M_ERLIP,R1        ; Set bit in R1 so as to return to caller.
                                07F4  1823 10$:
                50   0008'CF   3C  07F4  1824              MOVZWL  HIRT$W_IOST,R0          ; Indicate failure to caller.
                       05 50    E9  07F9  1825              BLBC    R0, 15$                 ; Branch if error already found.
                50   000A'CF   3C  07FC  1826              MOVZWL  HIRT$W_IOWORST, R0      ; Else, get worst case I/O status.
                                0801  1827 15$:    HIRT_SUBRETURN
                                0808  1828
                                0808  1829 REPLACE_CONNECT_FAILURE:                       ; Come here if CONNECTION failure
                                0808  1830                                                ;  anywhere in REPLACE logic.
                                0808  1831
                      F7F2'     31  0808  1832              BRW     DUTU$KILL_THIS_THREAD   ; Branch to kill this thread.
```

```
                     080E  1834          .SBTTL  DU$ONLINE_COMPLETE - Perform HIRT operations after ONLINE
                     080E  1835 ;++
                     080E  1836 ;
                     080E  1837 ; DU$ONLINE_COMPLETE - Perform HIRT operations after ONLINE
                     080E  1838 ;
                     080E  1839 ; Functional Description:
                     080E  1840 ;
                     080E  1841 ;       Complete bringing a unit ONLINE when it is attached to a controller
                     080E  1842 ;       that require HOST INITIATED dynamic bad block replacement.  This
                     080E  1843 ;       routine reads sector zero of the RCT to see if the disk went offline
                     080E  1844 ;       in the middle of bad block replacement.  If so the replacement is
                     080E  1845 ;       completed.
                     080E  1846 ;
                     080E  1847 ; Inputs:
                     080E  1848 ;
                     080E  1849 ;       R3      UCB address for the unit that being brought ONLINE
                     080E  1850 ;       R5      HIRT permanent CDRP address
                     080E  1851 ;
                     080E  1852 ; Implicit Inputs:
                     080E  1853 ;
                     080E  1854 ;       HIRT$L_SAVDCDRP         CDRP that describes the current operation
                     080E  1855 ;
                     080E  1856 ;       HIRT is owned by the current thread
                     080E  1857 ;       HIRT SUBSTACK is reset
                     080E  1858 ;
                     080E  1859 ; Outputs:
                     080E  1860 ;
                     080E  1861 ;       R0      Replacement status
                     080E  1862 ;       R1      Setting for CDRP$V_ERLIP
                     080E  1863 ;       R3      UCB address (unchanged)
                     080E  1864 ;
                     080E  1865 ;       R2, R4, R5 destroyed.
                     080E  1866 ;       All other registers preserved.
                     080E  1867 ;--
                     080E  1868
                     080E  1869
                     080E  1870 DU$ONLINE_COMPLETE::
                     080E  1871
                     080E  1872          HIRT_SUBSAVE                    ; Save return point on SUBSTACK.
                     0818  1873
                     0818  1874          ASSUME  HIRT$W_PG0CNTNT+2       EQ      HIRT$W_PG1CNTNT
004C'CF    01    CE  0818  1875          MNEGL   #1,HIRT$W_PG0CNTNT      ; Invalidate contents of pages 0 and 1.
                     081D  1876
                     081D  1877          ASSUME  HIRT$W_PG2CNTNT+2       EQ      HIRT$W_PG3CNTNT
0050'CF    01    CE  081D  1878          MNEGL   #1,HIRT$W_PG2CNTNT      ; Invalidate contents of pages 2 and 3.
                     0822  1879
                     0822  1880 ; Here we want to read sector 0 of the RCT into page 0 so as to be able to
                     0822  1881 ;       determine whether or not we went down in the middle of Dynamic Bad
                     0822  1882 ;       Block replacement.
                     0822  1883 ; Note we have NOT allocated a Message Buffer.  READ_RCT_BLOCK (via a call
                     0822  1884 ;       to BUILD_RCT_PACKET) will do it for us.
                     0822  1885
          50    7C   0822  1886          CLRQ    R0                      ; Indicate read sector 0 (R1) into
                     0824  1887                                          ;  page 0 (R0).
        019C    30   0824  1888          BSBW    READ_RCT_BLOCK          ; Read indicated sector into page.
     08 50    E8   0827  1889          BLBS    R0,20$                  ; LBS means successful read.
                     082A  1890          HIR_ERROR -                     ; Signal HIR error.
```

```
                                082A  1891                    step=1, func=ONLINE, -
                                082A  1892                    type=READ
                     FFB1   31  0832  1893            BRW     STEP18                  ; Goto deallocate and set RCT_FAILURE
                                0835  1894                                            ;  status code before returning.
                                0835  1895 20$:
                                0835  1896
                                0835  1897 ; Here we do a write of page 0 to sector 0 (all copies) to insure that we
                                0835  1898 ;       did not crash in the middle of an update of sector 0 and thereby
                                0835  1899 ;       get a set of inconsistent copies.
                                0835  1900
                          50    D4  0835  1901            CLRL    R0                      ; Rewrite page 0.
                        00CF    30  0837  1902            BSBW    WRITE_RCT_BLOCK         ; Write indicated sector from page.
                       0B 50    E8  083A  1903            BLBS    R0,30$                  ; LBS means successful write.
                                083D  1904            HIR_ERROR -                     ; Signal HIR error.
                                083D  1905                    step=2, func=ONLINE, -
                                083D  1906                    type=WRITE
                     FF9E   31  0845  1907            BRW     STEP18                  ; Goto deallocate and set RCT_FAILURE
                                0848  1908                                            ;  status code before returning.
                                0848  1909 30$:
                50   0024'CF    D0  0848  1910            MOVL    HIRT$L_PAGEOPTR,R0      ; R0 => sector 0 in memory.
                               B3  084D  1911            BITW    #RCT$M_RP1-             ; Test for phase 1 of replacement
                                084E  1912                    !RCT$M_RP2,-            ;  or phase 2.
             08 A0   C000 8F        084E  1913                    RCT$W_FLAGS(R0)
                           1B   12  0853  1914            BNEQ    40$                     ; NEQ implies that we were in the
                                0855  1915                                            ;  middle of replacement.
                                0855  1916            DEALLOC_MSG_BUF                 ; Else we deallocate the buffer
                          51    D4  0858  1917            CLRL    R1                      ; Prepare to return ERLOGIP bit if set.
                000E'CF     07    E5  085A  1918            BBCC    s^#HIRT$V_ERLOGIP, -    ; Branch around if clear and clear if
                                03  085F  1919                    HIRT$W_STS,35$          ;  already set.
                      51    04    AB  0860  1920            BISW    #CDRP$M_ERLIP,R1        ; Set bit in R1 so as to return to caller.
                                0863  1921 35$:
                      50    01    3C  0863  1922            MOVZWL  #SS$_NORMAL,R0          ; deallocate the RSPID, and we return
                                0866  1923                                            ;  to caller with a success status.
                                0866  1924            HIRT_SUBRETURN                  ; Return.
                                0870  1925 40$:
                        0C A0     D0  0870  1926            MOVL    RCT$L_LBN(R0),-         ; Restore LBN to replace to HIRT.
                      0018'CF           0873  1927                    HIRT$L_LBN
                      50    01    D0  0876  1928            MOVL    #1,R0                   ; Read into page 1,
                      51    01    D0  0879  1929            MOVL    #1,R1                   ; From RCT sector 1.
                         0144    30  087C  1930            BSBW    READ_RCT_BLOCK          ; Read RCT block.
                        0B 50    E8  087F  1931            BLBS    R0,50$                  ; LBS means successful read.
                                0882  1932            HIR_ERROR -                     ; Signal HIR error.
                                0882  1933                    step=3, func=ONLINE, -
                                0882  1934                    type=READ
                         FF59   31  088A  1935            BRW     STEP18                  ; Goto deallocate and set RCT_FAILURE
                                088D  1936                                            ;  status code before returning.
                                088D  1937 50$:
                50   0024'CF    D0  088D  1938            MOVL    HIRT$L_PAGEOPTR,R0      ; Again, R0 => sector 0 contents.
                                0892  1939
                               AA  0892  1940            BICW    #HIRT$M_FE-             ; Initialize incore bits.
                                0893  1941                    !HIRT$M_MATCH-
                                0893  1942                    !HIRT$M_EMPTYPE-
                                0893  1943                    !HIRT$M_RESCAN-
                                0893  1944                    !HIRT$M_RCTFULL,-
             000E'CF   007C 8F        0893  1945                    HIRT$W_STS
                                0899  1946
                           07   E1  0899  1947            BBC     #RCT$V_FE,-             ; Branch if no forced error.
```

```
                05 08 A0      089B  1948                    RCT$W_FLAGS(R0),60$
            000E'CF   04  AB  089E  1949            BISW    s^#HIRT$M_FE,HIRT$W_STS ; Set incore forec error bit.
                          08A3  1950 60$:
                   0F  E1   08A3  1951            BBC     #RCT$V_RP1,-            ; See if NOT in phase 1.
                03 08 A0      08A5  1952                    RCT$W_FLAGS(R0),70$
                    FB50  31  08A8  1953            BRW     STEP7                   ; Branch into step 7 to continue.
                          08AB  1954 70$:
                   0E  E0   08AB  1955            BBS     #RCT$V_RP2,-            ; Sanity check.
                04 08 A0      08AD  1956                    RCT$W_FLAGS(R0),75$
                          08B0  1957            BUG_CHECK       DISKCLASS,FATAL
                          08B4  1958 75$:
                   10 A0  D0  08B4  1959            MOVL    RCT$L_RBN(R0),-         ; Remember RBN that we had selected.
                 0058'CF      08B7  1960                    HIRT$L_RBN
                          08BA  1961
                   0D  E1   08BA  1962            BBC     #RCT$V_BR,-             ; See if we had had bad RBN.
                08 08 A0      08BC  1963                    RCT$W_FLAGS(R0),80$     ; Clear means no.
            000E'CF   08  AB  08BF  1964            BISW    s^#HIRT$M_MATCH, -      ; We set equivalent bit in core.
                          08C4  1965                    HIRT$W_STS
                   14 A0  D0  08C4  1966            MOVL    RCT$L_BAD_RBN(R0),-     ; Copy the old bad RBN.
                 005C'CF      08C7  1967                    HIRT$L_MATCHRBN
                          08CA  1968 80$:
                          08CA  1969
                          08CA  1970 ; Here figure out whether this is prime RBN.
                          08CA  1971
                     0309 30  08CA  1972            BSBW    HASH_LBN                ; Hash HIRT$L_LBN to produce values
                          08CD  1973                                                    ;  for HIRT$L_RCTBLOCK and HIRT$L_OFFSET.
          51  0068'CF   02  C3  08CD  1974            SUBL3   #2,HIRT$L_RCTBLOCK,R1   ; Subtract out RCT sectos 0 and 1.
              51   51   07  78  08D3  1975            ASHL    #7,R1,R1                ; R1 = relative block containing prime
                          08D7  1976                                                    ;  RBN descriptor * 128.
              51  006C'CF   C0  08D7  1977            ADDL    HIRT$L_OFFSET,R1        ; R1 = prime RBN.
            0058'CF   51   D1  08DC  1978            CMPL    R1,HIRT$L_RBN           ; See if this the one.
                      05   13  08E1  1979            BEQL    90$                     ; EQL implies yes.
                      10   A8  08E3  1980            BISW    #HIRT$M_EMPTYPE,-       ; Set bit meaning NOT prime RBN.
                    000E'CF       08E5  1981                    HIRT$W_STS
                          08E8  1982 90$:
                          08E8  1983
                          08E8  1984 ; Here read sector containing allocatable RBN into page 2.
                          08E8  1985
    51    0058'CF   F9 8F   78  08E8  1986            ASHL    #-7,HIRT$L_RBN,R1       ; R1 = relative RCT block containing
                          08EF  1987                                                    ;  this RBN descriptor.
                    51   02   C0  08EF  1988            ADDL    #2,R1                   ; Add in sectors 0 and 1.
                    50   02   D0  08F2  1989            MOVL    #2,R0                   ; Read into page 2.
                       00CB 30  08F5  1990            BSBW    READ_RCT_BLOCK          ; Go read RCT sector.
                     08 50  E8  08F8  1991            BLBS    R0,100$                 ; Branch on success.
                          08FB  1992
                          08FB  1993            HIR_ERROR -                     ; Signal HIR error.
                          08FB  1994                    step=4, func=ONLINE, -
                          08FB  1995                    type=READ
                       FEE0 31  0903  1996            BRW     STEP18                  ; Goto deallocate and set RCT_FAILURE
                          0906  1997                                                    ;  status code before returning.
                          0906  1998 100$:
                       FC59 31  0906  1999            BRW     STEP11
```

```
                     0909  2001          .SBTTL  WRITE_RCT_BLOCK - Write an RCT sector
                     0909  2002
                     0909  2003 ;+
                     0909  2004 ; WRITE_RCT_BLOCK - internal subroutine to write a block to a particular
                     0909  2005 ;       relative sector in each RCT copy.
                     0909  2006 ;
                     0909  2007 ; INPUTS:
                     0909  2008 ;       R0 =  page number from which we write the sector
                     0909  2009 ;       R3 => UCB
                     0909  2010 ;       R5 => CDRP
                     0909  2011 ;       HIRT$W_PGOCNTNT[R0] contains relative sector number to write
                     0909  2012 ;       CDRP$L_MSG_BUF contains an END PACKET to recycle
                     0909  2013 ;       CDRP$L_RSPID   contains an RSPID to recycle
                     0909  2014 ;
                     0909  2015 ; OUTPUTS:
                     0909  2016 ;       R0 - LBS indicates we were successful in writing at least one
                     0909  2017 ;            RCT.
                     0909  2018 ;       R0 - LBC indicates failure in all writes.
                     0909  2019 ;
                     0909  2020 ; SIDE EFFECTS:
                     0909  2021 ;       In this routine we use HIRT$W_IOST as the repository of the
                     0909  2022 ;       combined status of the writes that we execute.  In other words, if
                     0909  2023 ;       on finishing the writes, the low bit of HIRT$W_IOST is set,
                     0909  2024 ;       one or more of the writes was successful.
                     0909  2025 ;
                     0909  2026 ; NOTE:
                     0909  2027 ;       Since this subroutine is one of those that calls SCS routines which
                     0909  2028 ;       may fork, and since we may not leave anything permanent on the stack,
                     0909  2029 ;       the caller's return point is popped off the stack and pushed onto
                     0909  2030 ;       the HIRT SUBSTACK via use of the HIRT_SUBSAVE macro.  Return to the
                     0909  2031 ;       caller is effected by use of the HIRT_SUBRETURN macro.
                     0909  2032 ;
                     0909  2033
                     0909  2034 WRITE_RCT_BLOCK:
                     0909  2035
                     0909  2036          HIRT_SUBSAVE                            ; Save return point in SUBSTACK.
           0056'CF   50   B0 0913  2037          MOVW    R0,HIRT$W_PAGENO        ; Save input argument as to which page.
0054'CF    004C'CF40   B0 0918  2038          MOVW    HIRT$W_PGOCNTNT[R0],-   ; Also save input argument as to which
                     0920  2039                  HIRT$W_SECTORNO         ;  sector (page) to write.
  0008'CF    216C 8F   B0 0920  2040          MOVW    #SS$_BADRCT,HIRT$W_IOST ; Initialize combined status word.
             0010'CF   D4 0927  2041          CLRL    HIRT$L_LOOPCNT          ; Initialize loop counter. Note loop
                     092B  2042                                          ;  counter is longword even though we
                     092B  2043                                          ;  only use one byte since we MULL2
                     092B  2044                                          ;  with the counter in BUILD_RCT_PACKET.
        5F 68 A3     0D   E0 092B  2045          BBS     #UCB$V_MSCP_WRTP, -     ; If disk is software write protected
                     0930  2046                  UCB$W_DEVSTS(R3), 46$   ; branch around and reject.
                     0930  2047 20$:
             0100 8F   AA 0930  2048          BICW    #HIRT$M_RCTFE,-         ; Initialize flag each time thru loop.
             000E'CF      0934  2049                  HIRT$W_STS
             0010'CF   91 0937  2050          CMPB    HIRT$L_LOOPCNT,-        ; See if we are all done with all
             0000'C3      093B  2051                  UCB$B_DU_RCTCPYS(R3)    ;  RCT copies.
                   02   1F 093E  2052          BLSSU   30$                     ; LSSU implies NOT done.
                   6A   11 0940  2053          BRB     70$                     ; If done, branch around.
                     0942  2054 30$:
              50   0056'CF   3C 0942  2055          MOVZWL  HIRT$W_PAGENO,R0        ; R0 contains which page to map.
                     0161   30 0947  2056          BSBW    MAP_PAGE                ; Map page selected by R0.
                     094A  2057
```

```
                        094A  2058 ; Recycle the current END PACKET, the current RSPID and then prepare
                        094A  2059 ;       the MSCP packet to write page into the next RCT copy at relative
                        094A  2060 ;       sector of this copy.  All this is accomplished by BUILD_RCT_PACKET.
                        094A  2061
                 0108   30    094A  2062          BSBW    BUILD_RCT_PACKET        ; Routine fills most MSCP fields.
                        094D  2063                                                ;  Returns R2 => MSCP packet.
                   22   90    094D  2064          MOVB    #MSCP$K_OP_WRITE,-      ; Copy the WRITE opcode.
                 08 A2        094F  2065                  MSCP$B_OPCODE(R2)
              4000 8F   B0    0951  2066          MOVW    #MSCP$M_MD_COMP,-       ; Move in compare modifier to get a
                0A A2         0955  2067                  MSCP$W_MODIFIER(R2)     ;  write compare operation.
                              0957  2068
          000E'CF   08   E1   0957  2069          BBC     S^#HIRT$V_RCTFE, -      ; Bit clear says write WITHOUT force
                    06        095C  2070                  HIRT$W_STS,40$          ;  error.
              1000 8F   A8    095D  2071          BISW    #MSCP$M_MD_ERROR,-      ; Set force error modifier.
                0A A2         0961  2072                  MSCP$W_MODIFIER(R2)
                              0963  2073 40$:
                              0963  2074          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                              0966  2075          UNMAP                           ; Unmap page.
                2C A5   D4    0969  2076          CLRL    CDRP$L_LBUFH_AD(R5)     ; Indicate no mapping resources
                              096C  2077                                          ;  currently allocated.
             52 1C A5   D0    096C  2078          MOVL    CDRP$L_MSG_BUF(R5),R2   ; Refresh R2 => End message.
                              0970  2079
                   05   E1    0970  2080          BBC     #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                07 09 A2      0972  2081                  MSCP$B_FLAGS(R2),45$    ;  and branch around if not.
                0080 8F   A8  0975  2082          BISW    #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
                000E'CF       0979  2083                  HIRT$W_STS              ;  Have been generated.
                              097C  2084 45$:
                              097C  2085
                              097C  2086 ; See if write succeeded and if so set HIRT$W_IOST to success; otherwise
                              097C  2087 ;       do nothing.  In this way if one or more writes succeed, HIRT$W_IOST
                              097C  2088 ;       will have a success indication.
                              097C  2089
          000E'CF   08   E0   097C  2090          BBS     S^#HIRT$V_RCTFE, -      ; Branch around status update if
                    24        0981  2091                  HIRT$W_STS,60$          ;  we had force error.
                              0982  2092          IF_MSCP SUCCESS, then=50$, -    ; Branch if request was successful,
                              0982  2093                  status=R0               ; leaving MSCP status in R0.
             50   06   B1    098A  2094          CMPW    #MSCP$K_ST_WRTPR, R0    ; Check for write protected.
                    09   12    098D  2095          BNEQ    48$                     ; If NOT, some other error.
                              098F  2096 46$:
0008'CF   025C 8F   B0    098F  2097          MOVW    #SS$_WRITLCK,HIRT$W_IOST; Indicate why we couldn't write.
                    14   11    0996  2098          BRB     70$                     ; And branch around.
                              0998  2099 48$:
              0100 8F   A8    0998  2100          BISW    #HIRT$M_RCTFE,-         ; Set force error flag and
                000E'CF       099C  2101                  HIRT$W_STS
                    A1   11    099F  2102          BRB     30$                     ;  branch back to rewrite it.
                              09A1  2103 50$:
                    01   B0    09A1  2104          MOVW    S^#SS$_NORMAL,-         ; If success, remember it in static
                0008'CF       09A3  2105                  HIRT$W_IOST             ;  HIRT field.
                              09A6  2106 60$:
                0010'CF   D6  09A6  2107          INCL    HIRT$L_LOOPCNT          ; Increment loop counter.
                    B4   11   09AA  2108          BRB     20$                     ; And branch back to do next copy.
                              09AC  2109
                              09AC  2110 70$:                                     ; Here after we finish all RCT copies.
             50 0008'CF   3C  09AC  2111          MOVZWL  HIRT$W_IOST,R0          ; Return status to caller.
                 05 50    E8  09B1  2112          BLBS    R0, 75$                 ; Branch if successful.
          000A'CF   50    B0  09B4  2113          MOVW    R0, HIRT$W_IOWORST      ; Else, save "worst" error.
                              09B9  2114 75$:     HIRT_SUBRETURN                  ; Return to caller.
```

```
                                09C3  2116          .SBTTL  READ_RCT_BLOCK - Read an RCT sector
                                09C3  2117
                                09C3  2118 ;+
                                09C3  2119 ; READ_RCT_BLOCK - internal routine to read contents of a relative sector of
                                09C3  2120 ;       the RCT.
                                09C3  2121 ;
                                09C3  2122 ; INPUTS:
                                09C3  2123 ;       R0 =  page number of page into which we read the sector
                                09C3  2124 ;       R1 =  relative sector number to read
                                09C3  2125 ;       R3 => UCB
                                09C3  2126 ;       R5 => CDRP
                                09C3  2127 ;       CDRP$L_MSG_BUF contains an END PACKET to recycle
                                09C3  2128 ;       CDRP$L_RSPID   contains an RSPID to recycle
                                09C3  2129 ;
                                09C3  2130 ; OUTPUTS:
                                09C3  2131 ;       R0 - LBS indicates we were successful in the block from one of the
                                09C3  2132 ;            RCT copies.
                                09C3  2133 ;       R0 - LBC indicates failure in reading from all RCT copies.
                                09C3  2134 ;
                                09C3  2135 ; SIDE EFFECTS:
                                09C3  2136 ;       We use HIRT$W_IOST to temporarily save the status to be returned
                                09C3  2137 ;       to out caller.
                                09C3  2138 ;
                                09C3  2139 ; NOTE:
                                09C3  2140 ;       Since this subroutine is one of those that calls SCS routines which
                                09C3  2141 ;       may fork, and since we may not leave anything permanent on the stack,
                                09C3  2142 ;       the caller's return point is popped off the stack and pushed onto
                                09C3  2143 ;       the HIRT SUBSTACK via use of the HIRT_SUBSAVE macro.  Return to the
                                09C3  2144 ;       caller is effected by use of the HIRT_SUBRETURN macro.
                                09C3  2145 ;
                                09C3  2146
                                09C3  2147 READ_RCT_BLOCK:
                                09C3  2148
                                09C3  2149          HIRT_SUBSAVE                        ; Save return point in SUBSTACK.
          0056'CF    50    B0   09CD  2150          MOVW    R0,HIRT$W_PAGENO            ; Save input argument as to which page.
          0054'CF    51    B0   09D2  2151          MOVW    R1,HIRT$W_SECTORNO          ; Also save input argument as to which
                                09D7  2152                                              ;  sector to write.
     004C'CF40    01    AE   09D7  2153          MNEGW   #1,HIRT$W_PGOCNTNT[R0]      ; Invalidate page(R0) contents.
             0010'CF    D4   09DD  2154          CLRL    HIRT$L_LOOPCNT              ; Initialize loop counter.
                                09E1  2155 20$:
             0010'CF    91   09E1  2156          CMPB    HIRT$L_LOOPCNT,-            ; See if we are all done with all
             0000'C3                09E5  2157                  UCB$B_DU_RCTCPYS(R3)    ;  RCT copies.
                   09    1F   09E8  2158          BLSSU   30$                         ; LSSU implies NOT done.
                                09EA  2159
 0008'CF    216C    8F    B0   09EA  2160          MOVW    #SS$_BADRCT,HIRT$W_IOST     ; Pass failure to our caller.
                   48    11   09F1  2161          BRB     50$                         ; If done, branch around.
                                09F3  2162 30$:
       50    0056'CF    3C   09F3  2163          MOVZWL  HIRT$W_PAGENO,R0            ; R0 contains which page to map.
                00B0    30   09F8  2164          BSBW    MAP_PAGE                    ; Map page selected by R0.
                                09FB  2165
                                09FB  2166 ; Recycle the current END PACKET, the current RSPID and then prepare
                                09FB  2167 ;       the MSCP packet to read page from the next RCT copy at relative
                                09FB  2168 ;       sector of this copy.  All this is accomplished by BUILD_RCT_PACKET.
                                09FB  2169
                         58    10   09FB  2170          BSBB    BUILD_RCT_PACKET            ; Routine fills in most of MSCP packet.
                                09FD  2171                                              ;  Returns R2 => MSCP packet.
                         21    90   09FD  2172          MOVB    #MSCP$K_OP_READ,-           ; Copy the READ opcode since this field
```

```
                08 A2          09FF  2173                MSCP$B_OPCODE(R2)       ;  is not filled in by BUILD_RCT_PACKET.
          4000 8F    B0    0A01  2174          MOVW      #MSCP$M_MD_COMP,-       ; Move in compare modifier to get a
              0A A2          0A05  2175                    MSCP$W_MODIFIER(R2)     ;  compare operation.
                             0A07  2176
                             0A07  2177          SEND_MSCP_MSG                 ; Send message to the MSCP server.
                             0A0A  2178          UNMAP                         ; Unmap page.
              2C A5    D4    0A0D  2179          CLRL      CDRP$L_LBUFH_AD(R5)     ; Indicate no mapping resources
                             0A10  2180                                          ;  currently allocated.
          52  1C A5    D0    0A10  2181          MOVL      CDRP$L_MSG_BUF(R5),R2   ; Refresh R2 => End message.
                             0A14  2182
                05    E1    0A14  2183          BBC       #MSCP$V_EF_ERLOG,-      ; Test for error log message generated
                07 09 A2       0A16  2184                    MSCP$B_FLAGS(R2),35$    ;  and branch around if not.
              0080 8F    A8    0A19  2185          BISW      #HIRT$M_ERLOGIP,-       ; Else remember that error log messages
              000E'CF          0A1D  2186                    HIRT$W_STS              ;  Have been generated.
                             0A20  2187 35$:
                             0A20  2188
                             0A20  2189 ; See if read succeeded and if so we now have a valid copy of the sector so we
                             0A20  2190 ;       simply continue.  If we did not succeed we bump the loop counter and
                             0A20  2191 ;       go back to try and read the sector from the next (if any) RCT copy.
                             0A20  2192
                             0A20  2193          IF_MSCP SUCCESS, then=40$       ; Branch if request was successful.
              0010'CF    D6    0A26  2194          INCL      HIRT$L_LOOPCNT          ; Increment loop counter.
                   B5    11    0A2A  2195          BRB       20$                     ; And branch back to try next copy.
                             0A2C  2196
                             0A2C  2197 40$:                                     ; Here after we finish all RCT copies.
      50      0056'CF    3C    0A2C  2198          MOVZWL    HIRT$W_PAGENO,R0        ; R0 = page number into which we read.
              0054'CF    B0    0A31  2199          MOVW      HIRT$W_SECTORNO,-       ; Update contents of this page by
          004C'CF40              0A35  2200                    HIRT$W_PGOCNTNT[R0]     ;  remembering sector therein contained.
    0008'CF    01    B0    0A39  2201          MOVW      S^#SS$_NORMAL,HIRT$W_IOST; Indicate success.
                             0A3E  2202 50$:
      50      0008'CF    3C    0A3E  2203          MOVZWL    HIRT$W_IOST,R0          ; Return status to caller.
                   05 50    E8    0A43  2204          BLBS      R0, 55$                 ; Branch if successful.
    000A'CF       50    B0    0A46  2205          MOVW      R0, HIRT$W_IOWORST      ; Else, save "worst" error.
                             0A4B  2206 55$:     HIRT_SUBRETURN                ; Return to caller.
```

```
                     0A55  2208          .SBTTL  BUILD_RCT_PACKET - Recycle an MSCP end message
                     0A55  2209          .SBTTL  FILL_RCT_PACKET - Prepare an MSCP packet for an RCT transfer
                     0A55  2210
                     0A55  2211 ;+
                     0A55  2212 ; BUILD_RCT_PACKET - internal subroutine to recycle the current END PACKET
                     0A55  2213 ;                    and then to fall thru to
                     0A55  2214 ;
                     0A55  2215 ; FILL_RCT_PACKET - which prepares an MSCP packet to do an I/O transfer
                     0A55  2216 ;       to or from the RCT.
                     0A55  2217 ;
                     0A55  2218 ; INPUTS:
                     0A55  2219 ;       R3 => UCB
                     0A55  2220 ;       R5 => CDRP
                     0A55  2221 ;       CDRP$L_RSPID     contains a RSPID to re-cycle
                     0A55  2222 ;       CDRP$L_MSG_BUF   address of MSCP buffer to re-cycle or 0 (zero)
                     0A55  2223 ;                        0 (zero) means that we must here allocate an MSCP buffer
                     0A55  2224 ;       CDRP$L_BUFHNDL   contains 96 bit buffer handle
                     0A55  2225 ;       UCB$L_ABCNT      contains which RCT copy we are accessing
                     0A55  2226 ;       HIRT$W_SECTORNO contains which relative sector number in the RCT copy
                     0A55  2227 ;
                     0A55  2228 ; OUTPUTS:
                     0A55  2229 ;       R2 => MSCP PACKET
                     0A55  2230 ;       Registers R0 and R1 are modified
                     0A55  2231 ;
                     0A55  2232 ; NOTE:
                     0A55  2233 ;       Since BUILD_RCT_PACKET is one of those that calls SCS routines which
                     0A55  2234 ;       may fork, and since we may not leave anything permanent on the stack,
                     0A55  2235 ;       the caller's return point is popped off the stack upon entry to
                     0A55  2236 ;       this entrypoint and pushed onto the HIRT SUBSTACK via use of the
                     0A55  2237 ;       HIRT_SUBSAVE macro.  Upon return from those SCS routines, the caller's
                     0A55  2238 ;       return point is restored to the normal stack via use of the
                     0A55  2239 ;       HIRT_SUBUNSAVE macro.  all this is done prior to entrypoint
                     0A55  2240 ;       FILL_RCT_PACKET so that we may fall into this routine and then use
                     0A55  2241 ;       its RSB to return to our caller.
                     0A55  2242 ;
                     0A55  2243
                     0A55  2244 BUILD_RCT_PACKET:
                     0A55  2245
                     0A55  2246          HIRT_SUBSAVE                        ; Save return point on HIRT SUBSTACK.
                     0A5F  2247
                     0A5F  2248          ASSUME  MSCP$L_CMD_REF  EQ      0
 1C A5   D5          0A5F  2249          TSTL    CDRP$L_MSG_BUF(R5)          ; See if we need a Message Buffer.
    09   13          0A62  2250          BEQL    20$                         ; EQL means Buffer needed.
                     0A64  2251
                     0A64  2252          RECYCH_MSG_BUF                      ; Else Recycle END PACKET into MSCP buffer.
 09 50   E8          0A67  2253          BLBS    R0,30$                      ; LBS means allocation success.
  FD9E   31          0A6A  2254 10$:     BRW     REPLACE_CONNECT_FAILURE     ; Allocation failure means CONNECTION
                     0A6D  2255                                              ;  failure.
                     0A6D  2256 20$:
                     0A6D  2257          ALLOC_MSG_BUF                       ; Allocate a Message Buffer.
 F7 50   E9          0A70  2258          BLBC    R0,10$                      ; LBC means allocation failure.
                     0A73  2259 30$:
                     0A73  2260          HIRT_SUBUNSAVE                      ; Restore caller's return point.
                     0A7C  2261
                     0A7C  2262 FILL_RCT_PACKET:                             ; Alternate entry that only fills in packet.
                     0A7C  2263
                     0A7C  2264          INIT_MSCP_MSG ucb=(R3)              ; Initialize MSCP command packet.
```

```
                OC A2    0200 8F    3C    0A7F  2265          MOVZWL  #512, -                         ; Setup transfer byte count.
                                     0A85  2266                MSCP$L_BYTE_CNT(R2)
                      30 A5    7D    0A85  2267          MOVQ    CDRP$T_LBUFHNDL(R5),-             ; Copy 96 bit buffer handle.
                      10 A2          0A88  2268                MSCP$B_BUFFER(R2)                 ;                  "
                      38 A5    D0    0A8A  2269          MOVL    CDRP$T_LBUFHNDL+8(R5),-           ;                  "
                      18 A2          0A8D  2270                MSCP$B_BUFFER+8(R2)               ;
                                     0A8F  2271
                                     0A8F  2272 ; Calculate LBN of relative sector for this RCT copy.
                                     0A8F  2273 ;       It is done by multipling the number of RCT copies already written,
                                     0A8F  2274 ;       (contained in HIRT$L_LOOPCNT) by the size of an RCT copy (contained in
                                     0A8F  2275 ;       UCB$W_DU_RCTSIZE), adding in the LBN of the base of the first RCT copy
                                     0A8F  2276 ;       (UCB$L_DU_USIZE) and then adding in the relative sector number
                                     0A8F  2277 ;       passed to us when we were called (HIRT$W_SECTORNO).
                                     0A8F  2278
                50   0000'C3    3C    0A8F  2279          MOVZWL  UCB$W_DU_RCTSIZE(R3),R0 ; R0 contains size of one RCT copy.
                50   0010'CF    C4    0A94  2280          MULL2   HIRT$L_LOOPCNT,R0       ; R0 contains COPY# * COPYSIZE.
                50   0000'C3    C0    0A99  2281          ADDL    UCB$L_DU_USIZE(R3),R0   ; R0 contains LBN of base of this copy.
                51   0054'CF    3C    0A9E  2282          MOVZWL  HIRT$W_SECTORNO,R1      ; R1 contains input relative sector #.
                     50    51    C0    0AA3  2283          ADDL    R1,R0                   ; R0 contains LBN.
                  1C A2    50    D0    0AA6  2284          MOVL    R0,MSCP$L_LBN(R2)       ; Move LBN to MSCP packet.
                                     0AAA  2285
                                     05    0AAA  2286          RSB                             ; Return to caller.
```

```
                     0AAB  2288          .SBTTL  MAP_PAGE - Map a page for a transfer
                     0AAB  2289
                     0AAB  2290 ;+
                     0AAB  2291 ; MAP_PAGE - internal subroutine to map the page selected by R0.
                     0AAB  2292 ;
                     0AAB  2293 ; INPUTS:
                     0AAB  2294 ;       R0    contains the number of the page to map.
                     0AAB  2295 ;       R5 => CDRP
                     0AAB  2296 ;
                     0AAB  2297 ; OUTPUTS:
                     0AAB  2298 ;       CDRP$L_SVAPTE, CDRP$W_BOFF, CDRP$L_BCNT and set to page parameters.
                     0AAB  2299 ;
                     0AAB  2300 ;       CDRP$L_LBUFH_AD set to => CDRP$T_LBUFHNDL
                     0AAB  2301 ;
                     0AAB  2302 ;       Mapping resources allocated.
                     0AAB  2303
                     0AAB  2304 MAP_PAGE:
                     0AAB  2305          HIRT_SUBSAVE                        ; Save caller's return point on SUBSTACK.
CC A5  0034'CF40  D0 0AB5  2306          MOVL    HIRT$L_SVAPTE0[R0],-        ; Copy mapping date for relative page
                     0ABC  2307                  CDRP$L_SVAPTE(R5)           ;  to CDRP.
D0 A5  0044'CF40  B0 0ABC  2308          MOVW    HIRT$W_BOFF0[R0],-          ; Copy BOFF as well as SVAPTE.
                     0AC3  2309                  CDRP$W_BOFF(R5)
  D2 A5   0200 8F  3C 0AC3  2310          MOVZWL  #512,CDRP$L_BCNT(R5)        ; Finally copy BCNT for page.
                     0AC9  2311
        30 A5    9E 0AC9  2312          MOVAB   CDRP$T_LBUFHNDL(R5),-       ; Point CDRP field to local buffer
        2C A5       0ACC  2313                  CDRP$L_LBUFH_AD(R5)         ;  handle field.
                     0ACE  2314          MAP_IRP                             ; Map page.
                     0AD1  2315
                     0AD1  2316          HIRT_SUBRETURN                      ; Return to caller.
```

```
                     0ADB  2318          .SBTTL  SEARCH_RCT - Locate an available RBN
                     0ADB  2319
                     0ADB  2320 ;+
                     0ADB  2321 ; SEARCH_RCT - internal subroutine to search the RCT for an available RBN
                     0ADB  2322 ;       to allocate for the current failing LBN.  This routine is called from
                     0ADB  2323 ;       STEP9 of the replacement algorithm and is only done here as an
                     0ADB  2324 ;       internal subroutine to simplify the reading of that algorithm.
                     0ADB  2325 ;
                     0ADB  2326 ; INPUTS:
                     0ADB  2327 ;       R3 => UCB
                     0ADB  2328 ;       R5 => CDRP
                     0ADB  2329 ;       HIRT$L_LBN LBN that is failing
                     0ADB  2330 ;       UCB$W_DU_LBNPTRK number of LBNs on a track of this unit
                     0ADB  2331 ;       UCB$B_DU_RBNPTRK number of RBNs on a track of this unit
                     0ADB  2332 ;
                     0ADB  2333 ; OUTPUTS:
                     0ADB  2334 ;       R0 = SS$_NORMAL then:
                     0ADB  2335 ;               HIRT$L_RBN - new RBN selected to replace the failing LBN
                     0ADB  2336 ;               and HIRT$V_EMPTYPE clear means this is a primary RBN, else
                     0ADB  2337 ;               secondary RBN.
                     0ADB  2338 ;
                     0ADB  2339 ;               If HIRT$V_MATCH set this implies that the LBN which failed
                     0ADB  2340 ;                       had previously been replaced by an RBN which in
                     0ADB  2341 ;                       turn has failed.  This failing RBN is in
                     0ADB  2342 ;                       HIRT$L_MATCHRBN.
                     0ADB  2343 ;
                     0ADB  2344 ;       R0 = 0 then we could not find an allocatable RBN and HIRT$L_RBN is
                     0ADB  2345 ;               not valid. The cause of the failure to find an RBN is
                     0ADB  2346 ;               transmitted to the caller by:
                     0ADB  2347 ;
                     0ADB  2348 ;                       HIRT$V_RCTFULL set implies that the RCT on the disk
                     0ADB  2349 ;                               is full
                     0ADB  2350 ;                       HIRT$V_RCTFULL clear implies we had a read error on
                     0ADB  2351 ;                               some RCT sector.
                     0ADB  2352 ;-
                     0ADB  2353
                     0ADB  2354 SEARCH_RCT:
                     0ADB  2355
                     0ADB  2356          HIRT_SUBSAVE                    ; Save return on HIRT substack.
                     0AE5  2357
                00EE 30   0AE5  2358          BSBW    HASH_LBN                ; Hash LBN value in HIRT$L_LBN returning
                     0AE8  2359                                            ;  HIRT$L_RCTBLOCK and HIRT$L_OFFSET.
                     0AE8  2360
0068'CF         D0   0AE8  2361          MOVL    HIRT$L_RCTBLOCK,-       ; And remember the starting sector
0064'CF              0AEC  2362                  HIRT$L_STARTBLK         ;  number in static storage.
                     0AEF  2363
                     0AEF  2364 ; Here we initialize a few bits.
                     0AEF  2365
                AA   0AEF  2366          BICW    #HIRT$M_MATCH-          ; Initialize the following flags.
                     0AF0  2367                  !HIRT$M_EMPTYPE-        ;  Match set implies valid MATCHRBN,
                     0AF0  2368                  !HIRT$M_RESCAN-         ;  EMPTYPE set implies secondary RBN,
                     0AF0  2369                  !HIRT$M_RCTFULL,-       ;  Rescan implies reached Nulls,
000E'CF    0078 8F   0AF0  2370                  HIRT$W_STS              ;  and RCTFULL means the RCT is full.
                     0AF6  2371
                     0AF6  2372 ; Here we prepare to read the RCT sector containing the primary RBN descriptor.
                     0AF6  2373
                50   02   D0  0AF6  2374          MOVL    #2,R0                   ; Prepare to read into page #2.
```

```
                51   0068'CF    D0  0AF9  2375          MOVL    HIRT$L_RCTBLOCK,R1      ; And we read this relative sector #.
                       FEC2     30  0AFE  2376          BSBW    READ_RCT_BLOCK          ; Subroutine does read.
                     77 50      E9  0B01  2377          BLBC    R0,SEARCH_RTN           ; LBC implies read failure.
                                    0B04  2378
                                    0B04  2379  ; Here we scan the RCT sector containing the primary RBN descriptor.  The
                                    0B04  2380  ;       method of scanning is to scan outward from the primary RBN descriptor.
                                    0B04  2381
                                    0B04  2382  10$:
                         52     D4  0B04  2383          CLRL    R2                      ; Set up delta.
                                    0B06  2384  20$:
           51   006C'CF   52     C1  0B06  2385          ADDL3   R2,HIRT$L_OFFSET,R1    ; R1 = next entry to test in first RCT
                                    0B0C  2386                                          ;  sector to scan.
                         0F     19  0B0C  2387          BLSS    40$                     ; LSS implies invalid offset into page.
                                    0B0E  2388  30$:
    0000007F 8F          51     D1  0B0E  2389          CMPL    R1,#127                 ; See if we are within sector page.
                         06     14  0B15  2390          BGTR    40$                     ; GTR implies no, out of bounds, go
                                    0B17  2391                                          ;  to increment delta.
                       006B     30  0B17  2392          BSBW    TEST_RCT_ENTRY          ; If in bounds, go test RCT entry.
                     5E 50      E8  0B1A  2393          BLBS    R0,SEARCH_RTN           ; LBS implies success.
                                    0B1D  2394  40$:
                   52   52      CE  0B1D  2395          MNEGL   R2,R2                   ; Negate delta.
                        E4     19  0B20  2396          BLSS    20$                     ; Branch to try again if negative.
                        52     D6  0B22  2397          INCL    R2                      ; Else increment delta.
    00000080 8F         52     D1  0B24  2398          CMPL    R2,#128                 ; See if delta too big.
                        D9     19  0B2B  2399          BLSS    20$                     ; LSS implies not too big.
                                    0B2D  2400                                          ; Else we fall thru to try next sector.
                                    0B2D  2401  NEXT:
                 0068'CF    D6  0B2D  2402          INCL    HIRT$L_RCTBLOCK         ; Increment RCT sector to scan.
                                    0B31  2403  10$:
                 0068'CF    D1  0B31  2404          CMPL    HIRT$L_RCTBLOCK,-       ; See if we are all done with search.
                 0064'CF        0B35  2405                  HIRT$L_STARTBLK
                        38  13  0B38  2406          BEQL    SEARCH_FAIL             ; EQL means that we are finished.
                                    0B3A  2407
                   50   02     D0  0B3A  2408          MOVL    #2,R0                   ; Prepare to read into page 2.
             51    0068'CF    D0  0B3D  2409          MOVL    HIRT$L_RCTBLOCK,R1      ; And to read this sector.
                     FE7E     30  0B42  2410          BSBW    READ_RCT_BLOCK          ; Go to read sector into page.
                   33 50      E9  0B45  2411          BLBC    R0,SEARCH_RTN           ; LBC implies read failure
                                    0B48  2412
                        1F     E1  0B48  2413          BBC     #RCT$V_NULL,-           ; Before linear scan of this sector,
                OF 002C'DF          0B4A  2414                  @HIRT$L_PAGE2PTR,20$    ;  see if we are beyond RCT.
          0068'CF    02     D0  0B4E  2415          MOVL    #2,HIRT$L_RCTBLOCK      ; Here beyond RCT.  Wrap to start and
          000E'CF    05     E3  0B53  2416          BBCS    s^#HIRT$V_RESCAN, -     ;  go back to search some more after
                        D8          0B58  2417                  HIRT$W_STS,10$          ;  setting bit that says we have wrapped.
                                    0B59  2418          BUG_CHECK       DISKCLASS,FATAL ; Impossible situation.
                                    0B5D  2419  20$:
                         52     D4  0B5D  2420          CLRL    R2                      ; Clear loop index register.
                                    0B5F  2421  30$:
                   51    52     D0  0B5F  2422          MOVL    R2,R1                   ; Pass RCT entry of interest to routine.
                       0020     30  0B62  2423          BSBW    TEST_RCT_ENTRY          ; Call subroutine to test entry.
                     13 50      E8  0B65  2424          BLBS    R0,SEARCH_RTN           ; LBS means we have the RBN, go from loop.
EF 52      00000080 8F         F2  0B68  2425          AOBLSS  #128,R2,30$             ; If we return here, (entry not avail.)
                                    0B70  2426                                          ;  then loop back after incrementing R2.
                         BB     11  0B70  2427          BRB     NEXT                    ; If we fall thru, goto NEXT sector.
                                    0B72  2428
                                    0B72  2429  SEARCH_FAIL:
                         50     D4  0B72  2430          CLRL    R0                      ; Indicate failure to caller and
                       0040 8F    A8  0B74  2431          BISW    #HIRT$M_RCTFULL,-       ;  indicate reason for failure.
```

```
                 000E'CF        0B78  2432                      HIRT$W_STS
                               0B78  2433 SEARCH_RTN:
                               0B78  2434         HIRT_SUBRETURN                  ; Return to caller.
```

```
                                0B85  2436          .SBTTL  TEST_RCT_ENTRY - Test for allocated RBN
                                0B85  2437
                                0B85  2438 ;+
                                0B85  2439 ; TEST_RCT_ENTRY - internal subroutine called to test an RCT entry to see
                                0B85  2440 ;       if it represents an allocatable RBN or if it is already allocated.
                                0B85  2441 ;
                                0B85  2442 ; INPUTS:
                                0B85  2443 ;       R1 = index of RCT entry.
                                0B85  2444 ;
                                0B85  2445 ; OUTPUTS:
                                0B85  2446 ;       R0 = success code.
                                0B85  2447 ;          = SS$_NORMAL then HIRT$L_RBN is set to the RBN associated with the
                                0B85  2448 ;               RCT entry defined by HIRT$L_RCTBLOCK and R1 (entry index).
                                0B85  2449 ;
                                0B85  2450 ;          = 0 implies that the entry is not allocatable.  In in addition if
                                0B85  2451 ;               the RBN is currently allocated and if it is allocated to this
                                0B85  2452 ;               LBN (i.e. to HIRT$L_LBN), then HIRT$V_MATCH is set in
                                0B85  2453 ;               HIRT$W_STS and the RBN associated with the current entry is
                                0B85  2454 ;               stored in HIRT$L_MATCHRBN.
                                0B85  2455 ;
                                0B85  2456
                                0B85  2457
                                0B85  2458 TEST_RCT_ENTRY:
                      52    DD  0B85  2459          PUSHL   R2                      ; Save register.
50      0068'CF       02    C3  0B87  2460          SUBL3   #2,HIRT$L_RCTBLOCK,R0   ; R0 = found sector without bias of 2.
        50    50      07    78  0B8D  2461          ASHL    #7,R0,R0                ; Multiply by 128.
              50      51    C0  0B91  2462          ADDL    R1,R0                   ; R0 = RBN associated with this entry.
                                0B94  2463
        52    002C'CF       D0  0B94  2464          MOVL    HIRT$L_PAGE2PTR,R2      ; R2 => page 2, which contains sector.
                6241    D5    0B99  2465          TSTL    (R2)[RT]                ; Test contents of current entry.
                  0A    12    0B9C  2466          BNEQ    10$                     ; NEQ implies that it is not available.
0058'CF         50    D0    0B9E  2467          MOVL    R0,HIRT$L_RBN           ; Save RBN of this entry in HIRT.
        50      01    3C    0BA3  2468          MOVZWL  #SS$_NORMAL,R0          ; Set success code.
                2A    11    0BA6  2469          BRB     40$                     ; And branch to return to caller.
                                0BA8  2470 10$:
   52     6241    DE    0BA8  2471          MOVAL   (R2)[R1],R2              ; R2 => entry of interest.
                10    A8    0BAC  2472          BISW    #HIRT$M_EMPTYPE,-        ; Set bit meaning any find will now
          000E'CF             0BAE  2473                  HIRT$W_STS               ;  have to be a secondary RBN.
 1B 62      1D    E1    0BB1  2474          BBC     #RCT$V_ALLOCATED,(R2),30$; If clear, then unusable RBN.
                                0BB5  2475
                00    EF    0BB5  2476          EXTZV   #RCT$V_LBN,-             ; If allocated, see if for this LBN.
52      62      1C          0BB7  2477                  #RCT$S_LBN,(R2),R2       ; R2 = LBN for this RBN.
0018'CF         52    D1    0BBA  2478          CMPL    R2,HIRT$L_LBN            ; See if this LBN.
                0F    12    0BBF  2479          BNEQ    30$                      ; NEQ means not for this LBN.
000E'CF         03    E3    0BC1  2480          BBCS    s^#HIRT$V_MATCH, -       ; Set bit that means we have a match.
                04          0BC6  2481                  HIRT$W_STS,20$
                            0BC7  2482          BUG_CHECK       DISKCLASS,FATAL ; Impossible situation.
                            0BCB  2483 20$:
005C'CF         50    D0    0BCB  2484          MOVL    R0,HIRT$L_MATCHRBN       ; Save RBN that matched.
                            0BD0  2485 30$:
                50    D4    0BD0  2486          CLRL    R0                       ; Failure to find allocatable RBN.
                            0BD2  2487 40$:
              52 8ED0       0BD2  2488          POPL    R2                       ; Restore register.
                    05      0BD5  2489          RSB                              ; Return to caller.
```

```
                        0BD6  2491          .SBTTL  HASH_LBN - Hash an LBN into a RCT block and an offset
                        0BD6  2492
                        0BD6  2493 ;+
                        0BD6  2494 ; HASH_LBN - internal routine to hash HIRT$L_LBN giving HIRT$L_RCTBLOCK and
                        0BD6  2495 ;       HIRT$L_OFFSET.
                        0BD6  2496 ;
                        0BD6  2497 ; INPUTS:
                        0BD6  2498 ;
                        0BD6  2499 ;       R3 => UCB
                        0BD6  2500 ;       HIRT$L_LBN
                        0BD6  2501 ;
                        0BD6  2502 ; OUTPUTS:
                        0BD6  2503 ;
                        0BD6  2504 ;       HIRT$L_RCTBLOCK = RCT sector containing prime RBN descriptor for this
                        0BD6  2505 ;                         LBN
                        0BD6  2506 ;       HIRT$L_OFFSET = offset of prime RBN descriptor in sector.
                        0BD6  2507 ;
                        0BD6  2508 ; SIDE EFFECTS:
                        0BD6  2509 ;
                        0BD6  2510 ;       Registers R0 an R1 altered.
                        0BD6  2511 ;
                        0BD6  2512
                        0BD6  2513 HASH_LBN:
                        0BD6  2514
     50   0000'C3   3C  0BD6  2515          MOVZWL  UCB$W_DU_LBNPTRK(R3),R0 ; R0 contains LBNs per track.
51   0018'CF   50   C7  0BDB  2516          DIVL3   R0,HIRT$L_LBN,R1        ; R1 =  QUO(LBN/(LBNs per track)).
                        0BE1  2517
     50   0000'C3   9A  0BE1  2518          MOVZBL  UCB$B_DU_RBNPTRK(R3),R0 ; R0 = RBNs per track.
          50   51   C4  0BE6  2519          MULL    R1,R0                   ; R0 = (RBNs per)*QUO(LBN/(LBNs per))
               51   D4  0BE9  2520          CLRL    R1                      ; Clear high order part of dividend.
                        0BE6  2521
 50   00000080 8F   7B  0BEB  2522          EDIV    #128,R0,-               ; Divide result by 128 giving the
                        0BF2  2523                  HIRT$L_RCTBLOCK,-       ;  quotient and the
006C'CF   0068'CF       0BF2  2524                  HIRT$L_OFFSET           ;  remainder.
                        0BF8  2525
     0068'CF   02   C0  0BF8  2526          ADDL    #2,HIRT$L_RCTBLOCK      ; Add in sector 0 and sector 1.
               05   0BFD  2527          RSB                             ; Return to caller
```

```
                     0BFE  2529          .SBTTL  DU$HIR_ERROR - Process error encountered during HIRT processing
                     0BFE  2530 ;++
                     0BFE  2531 ;
                     0BFE  2532 ; DU$HIR_ERROR - Process error encountered during HIRT processing
                     0BFE  2533 ;
                     0BFE  2534 ; Functional Description:
                     0BFE  2535 ;
                     0BFE  2536 ;       This routine performs any operations necessary to inform the world
                     0BFE  2537 ;       that an error was encountered during HIRT processing.  It is invoked
                     0BFE  2538 ;       via the HIR_ERROR macro.
                     0BFE  2539 ;
                     0BFE  2540 ;       Currently, the error processing consists of broadcasting a message to
                     0BFE  2541 ;       OPA0.  The general text of the error message is:
                     0BFE  2542 ;
                     0BFE  2543 ;               %<devnam> encountered a <type> error in <func> step <n>
                     0BFE  2544 ;
                     0BFE  2545 ;       Where:
                     0BFE  2546 ;
                     0BFE  2547 ;               <devnam>        is the device name
                     0BFE  2548 ;               <type>          is one of READ / WRITE / RCT FULL
                     0BFE  2549 ;               <func>          is one of REPLACE / ONLINE
                     0BFE  2550 ;               <n>             is a number giving the failing step in the
                     0BFE  2551 ;                               replacement algorithm
                     0BFE  2552 ;
                     0BFE  2553 ; Inputs:
                     0BFE  2554 ;
                     0BFE  2555 ;       R0      A parameter giving the <type>, <func>, and <n> values above
                     0BFE  2556 ;       R3      UCB address
                     0BFE  2557 ;
                     0BFE  2558 ; Outputs:
                     0BFE  2559 ;
                     0BFE  2560 ;       R0 is destroyed
                     0BFE  2561 ;       All other registers are preserved.
                     0BFE  2562 ;--
                     0BFE  2563
                     0BFE  2564 DU$HIR_ERROR:
                     0BFE  2565
    00FE  8F    BB   0BFE  2566          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7>       ; Save some registers.
 5E   88  AE    9E   0C02  2567          MOVAB   -HIRER$K_MSGSIZE(SP), SP        ; Make message space on stack.
    56    5E    D0   0C06  2568          MOVL    SP, R6                          ; Save base of message space.
    57    50    D0   0C09  2569          MOVL    R0, R7                          ; Copy error parameter.
                     0C0C  2570
                     0C0C  2571          ; Form device name.
    55    53    D0   0C0C  2572          MOVL    R3, R5                          ; Move UCB address.
    51    56    D0   0C0F  2573          MOVL    R6, R1                          ; Setup buffer address.
    81    25    90   0C12  2574          MOVB    #^A/%/, (R1)+                   ; Insert percent sign.
    50    12    D0   0C15  2575          MOVL    #HIRER$K_DEVNAMSIZ, R0          ; Setup buffer size.
    54    01    CE   0C18  2576          MNEGL   #1, R4                          ; Setup formation code.
00000000'GF   16    0C1B  2577          JSB     G^IOC$CVT_DEVNAM                ; Get device name for UCB.
53  56    51    C1   0C21  2578          ADDL3   R1, R6, R3                      ; Init working buffer pointer.
          53    D6   0C25  2579          INCL    R3                              ; Adjust for percent sign.
                     0C27  2580
                     0C27  2581          ; Copy first fixed segment.
51  0034'CF    9E   0C27  2582          MOVAB   HIR_ERR_SEG1, R1                ; Get string address.
          68    10   0C2C  2583          BSBB    COPY_ASCIC                      ; Copy string.
                     0C2E  2584
                     0C2E  2585          ; Insert proper <type> segment.
```

```
                52   57   04   08   EF  0C2E  2586                EXTZV   #HIRER$V_TYPE, #HIRER$S_TYPE, R7, R2    ; Get type number.
                51   0000'CF   9E  0C33  2587                MOVAB   HIR_ERR_TYPES, R1                       ; Get error types strings base.
                     50   81   9A  0C38  2588 32$:           MOVZBL  (R1)+, R0                               ; Get length of this message.
                     51   50   C0  0C3B  2589                ADDL    R0, R1                                  ; Point to next message.
                       F7   52   F5  0C3E  2590                SOBGTR  R2, 32$                                 ; Loop till message located.
                            53   10  0C41  2591                BSBB    COPY_ASCIC                              ; Copy <type> string.
                                     0C43  2592
                                     0C43  2593                ; Copy second fixed segment.
                51   0044'CF   9E  0C43  2594                MOVAB   HIR_ERR_SEG2, R1                        ; Get string address.
                            4C   10  0C48  2595                BSBB    COPY_ASCIC                              ; Copy string.
                                     0C4A  2596
                                     0C4A  2597                ; Insert proper <func> segment.
                51   0025'CF   9E  0C4A  2598                MOVAB   HIR_ERR_REPLACE, R1                     ; Assume REPLACE.
                  05 57   0C   E1  0C4F  2599                BBC     #HIRER$V_ONLINE, R7, 45$                ; Branch if not ONLINE.
                51   002D'CF   9E  0C53  2600                MOVAB   HIR_ERR_ONLINE, R1                      ; Else, get ONLINE.
                            3C   10  0C58  2601 45$:         BSBB    COPY_ASCIC                              ; Copy <func> string.
                                     0C5A  2602
                                     0C5A  2603                ; Copy third fixed segment.
                51   004F'CF   9E  0C5A  2604                MOVAB   HIR_ERR_SEG3, R1                        ; Get string address.
                            35   10  0C5F  2605                BSBB    COPY_ASCIC                              ; Copy string.
                                     0C61  2606
                                     0C61  2607                ; Convert two digits of <n> and insert them.
                                     0C61  2608                ASSUME  HIRER$V_STEP EQ 0
                                     0C61  2609                ASSUME  HIRER$S_STEP EQ 8
                     50   57   9A  0C61  2610                MOVZBL  R7, R0                                  ; Get step number.
                  50   63   8F   91  0C64  2611                CMPB    #99, R0                                 ; Is number to big?
                            0F   15  0C68  2612                BLEQ    60$                                     ; Branch if number to big.
                            51   D4  0C6A  2613                CLRL    R1                                      ; Quadword extend number.
          51    50   50   0A   7B  0C6C  2614                EDIV    #10, R0, R0, R1                         ; Split digits.
                83   50   30   81  0C71  2615                ADDB3   #^A/0/, R0, (R3)+                       ; Insert tens digit.
                83   51   30   81  0C75  2616                ADDB3   #^A/0/, R1, (R3)+                       ; Insert units digit.
                                     0C79  2617
                                     0C79  2618 60$:          ; Compute message size and broadcast message to OPA0:.
                     52   56   D0  0C79  2619                MOVL    R6, R2                                  ; Setup base message address.
                51   53   52   C3  0C7C  2620                SUBL3   R2, R3, R1                              ; Setup message size.
          55     00000000'GF   9E  0C80  2621                MOVAB   G^OPA$UCB0, R5                          ; Get OPA0 UCB address.
                 00000000'GF   16  0C87  2622                JSB     G^IOC$BROADCAST                         ; Broadcast message.
                                     0C8D  2623
                5E    48   AE   9E  0C8D  2624                MOVAB   HIRER$K_MSGSIZE(SP), SP                 ; Clear message from stack.
                   00FE   8F   BA  0C91  2625                POPR    #^M<R1,R2,R3,R4,R5,R6,R7>               ; Restore saved registers.
                            05   0C95  2626                RSB                                             ; Exit.
                                     0C96  2627
                                     0C96  2628 ;**
                                     0C96  2629 ;
                                     0C96  2630 ; Routine to copy ASCIC string to buffer.
                                     0C96  2631 ;
                                     0C96  2632 ; Inputs:
                                     0C96  2633 ;
                                     0C96  2634 ;       R1      ASCIC string address
                                     0C96  2635 ;       R3      buffer address
                                     0C96  2636 ;
                                     0C96  2637 ; Outputs:
                                     0C96  2638 ;
                                     0C96  2639 ;       R3      updated buffer address (complements of MOVC3)
                                     0C96  2640 ;
                                     0C96  2641 ;       R0 through R5 are altered.
                                     0C96  2642 ;       All other registers are preserved.
```

```
                     0C96  2643 ;--
                     0C96  2644
                     0C96  2645 COPY_ASCIC:
      50    B1   9A  0C96  2646         MOVZBL  (R1)+, R0               ; Get string size.
63    61    50   28  0C99  2647         MOVC3   R0, (R1), (R3)          ; Copy string.
                05   0C9D  2648         RSB
                     0C9E  2649
                     0C9E  2650
                     0C9E  2651
                     0C9E  2652         .END
```

```
ALLOC_POOL                      000000B4 R      02
ATE_MSCPCODE                    00000002
ATE_OFFSET                      00000000
ATE_SSCODE                      00000003
BIT...                        = 0000000D
BUG$_DISKCLASS                  ********   X   02
BUILD_RCT_PACKET                00000A55 R      02
CDDB$A_2PFKB                    00000174
CDDB$A_DAPCDRP                  00000194
CDDB$A_DAPIRP                   00000134
CDDB$A_PRMCDRP                  000000D0
CDDB$A_PRMIRP                   00000070
CDDB$B_SUBTYPE                = 0000000B
CDDB$B_TYPE                   = 0000000A
CDDB$K_LENGTH                 = 00000070
CDDB$L_CANCLQBL                 000000B4
CDDB$L_CANCLQFL                 000000B0
CDDB$L_CDT                      000000F4
CDDB$L_DAPCDT                   000001B8
CDDB$L_DAPUCB                   00000150
CDDB$L_PRMUCB                   0000008C
CDDB$L_SAVED_PC               = 00000044
CDDB$W_SIZE                   = 00000008
CDRP$B_CARCON                 = FFFFFFDC
CDRP$B_CD_TYPE                = 0000000A
CDRP$B_EFN                    = FFFFFFC2
CDRP$B_IRP_TYPE               = FFFFFFAA
CDRP$B_PRI                    = FFFFFFC3
CDRP$B_RMOD                   = FFFFFFAB
CDRP$L_ABCNT                  = FFFFFFE0
CDRP$L_ARB                    = FFFFFFF8
CDRP$L_AST                    = FFFFFFB0
CDRP$L_ASTPRM                 = FFFFFFB4
CDRP$L_BCNT                   = FFFFFFD2
CDRP$L_CDT                    = 00000024
CDRP$L_DIAGBUF                = FFFFFFEC
CDRP$L_DUTUFLAGS              = 00000040
CDRP$L_EXTEND                 = FFFFFFF4
CDRP$L_FR3                    = 00000010
CDRP$L_FR4                    = 00000014
CDRP$L_IOQBL                  = FFFFFFA4
CDRP$L_IOQFL                  = FFFFFFA0
CDRP$L_IOSB                   = FFFFFFC4
CDRP$L_IOST1                  = FFFFFFD8
CDRP$L_IOST2                  = FFFFFFDC
CDRP$L_JNL_SEQNO              = FFFFFFE8
CDRP$L_LBUFH_AD               = 0000002C
CDRP$L_MEDIA                  = FFFFFFD8
CDRP$L_MSG_BUF                = 0000001C
CDRP$L_OBCNT                  = FFFFFFE4
CDRP$L_PID                    = FFFFFFAC
CDRP$L_RSPID                  = 00000020
CDRP$L_RWCPTR                 = 00000028
CDRP$L_SAVD_RTN               = 00000018
CDRP$L_SEGVBN                 = FFFFFFE8
CDRP$L_SEQNUM                 = FFFFFFF0
CDRP$L_SVAPTE                 = FFFFFFCC
CDRP$L_TT_TERM                = FFFFFFDC
CDRP$L_UBARSRCE               = 0000003C
CDRP$L_UCB                    = FFFFFFBC
CDRP$L_WIND                   = FFFFFFB8
CDRP$M_ERLIP                  = 00000004
CDRP$M_HIRT                   = 00000010
CDRP$Q_NT_PRVMSK              = FFFFFFE0
CDRP$T_LBUFHNDL               = 00000030
CDRP$V_PERM                   = 00000003
CDRP$W_ABCNT                  = FFFFFFE0
CDRP$W_BCNT                   = FFFFFFD2
CDRP$W_BOFF                   = FFFFFFD0
CDRP$W_CDRPSIZE               = 00000008
CDRP$W_CHAN                   = FFFFFFC8
CDRP$W_DUTUCNTR               = 00000044
CDRP$W_FUNC                   = FFFFFFC0
CDRP$W_IRP_SIZE               = FFFFFFA8
CDRP$W_OBCNT                  = FFFFFFE4
CDRP$W_STS                    = FFFFFFCA
COPY_ASCIC                      00000C96 R      02
DEVNAMSIZ                     = 00000012
DU$CANCEL_FROM_HIRT             00000229 RG     02
DU$DISCONNECT_HIRT              000002A9 RG     02
DU$HIR_ERROR                    00000BFE R      02
DU$INIT_HIRT                    00000000 RG     02
DU$LOCK_HIRT                    000000EF RG     02
DU$ONLINE_COMPLETE              0000080E RG     02
DU$REPLACE_LBN                  0000031D RG     02
DU$RSTRTQ_HIRT_CDRP             000002E0 RG     02
DU$TEST_HIRT_RQAITCNT           00000205 RG     02
DU$UNLOCK_HIRT                  00000178 RG     02
DUTU$INIT_MSCP_MSG_UNIT         ********   X    02
DUTU$INSERT_RESTARTQ            ********   X    02
DUTU$KILL_THIS_THREAD           ********   X    02
DUTU$L_CDDB_LISTHEAD            00000000
DUTU$POST_CDRP                  ********   X    02
DUTU$SEND_MSCP_MSG              ********   X    02
DUTU$TEST_CANCEL_CDRP           ********   X    02
DYN$C_CDRP                    = 00000039
DYN$C_CD_BBRPG                = 00000002
DYN$C_CLASSDRV                = 00000064
DYN$C_IRP                     = 0000000A
END_INIT_HIRT                   000000B1 R      02
EXE$ALONONPAGED                 ********   X    02
EXE$FORK_WAIT                   ********   X    02
FILL_RCT_PACKET                 00000A7C R      02
FIXEDSIZ                      = 00000023
FKB$K_LENGTH                  = 00000018
FKB$L_FPC                     = 0000000C
FKB$L_FQBL                    = 00000004
FKB$L_FQFL                    = 00000000
FKB$L_FR3                     = 00000010
FUNCSIZ                       = 00000007
GRANT_HIRT                      00000120 R      02
HASH_LBN                        00000BD6 R      02
HIRER$K_DEVNAMSIZ             = 00000012
HIRER$K_MSGSIZE               = 00000048
```

```
HIRER$K_RCTFULL = 00000003
HIRER$K_READ = 00000001
HIRER$K_REPFAIL = 00000004
HIRER$K_WRITE = 00000002
HIRER$M_ONLINE = 00001000
HIRER$M_REPLACE = 00000000
HIRER$M_STEP = 000000FF
HIRER$M_TYPE = 00000F00
HIRER$S_STEP = 00000008
HIRER$S_TYPE = 00000004
HIRER$V_ONLINE = 0000000C
HIRER$V_STEP = 00000000
HIRER$V_TYPE = 00000008
HIRT$K_SUBSTKLN = 00000005
HIRT$L_BADRBND 00000060 R 03
HIRT$L_CDRP 00000020 R 03
HIRT$L_LBN 00000018 R 03
HIRT$L_LOOPCNT 00000010 R 03
HIRT$L_MATCHRBN 0000005C R 03
HIRT$L_OFFSET 0000006C R 03
HIRT$L_OWNUCB 00000014 R 03
HIRT$L_PAGE0PTR 00000024 R 03
HIRT$L_PAGE1PTR 00000028 R 03
HIRT$L_PAGE2PTR 0000002C R 03
HIRT$L_PAGE3PTR 00000030 R 03
HIRT$L_RBN 00000058 R 03
HIRT$L_RCTBLOCK 00000068 R 03
HIRT$L_RPLQFL 00000000 R 03
HIRT$L_RPLQTP 00000004 R 03
HIRT$L_SAVDCDRP 0000001C R 03
HIRT$L_STARTBLK 00000064 R 03
HIRT$L_STKPTR 00000070 R 03
HIRT$L_SUBSTACK 00000074 R 03
HIRT$L_SVAPTE0 00000034 R 03
HIRT$L_SVAPTE1 00000038 R 03
HIRT$L_SVAPTE2 0000003C R 03
HIRT$L_SVAPTE3 00000040 R 03
HIRT$M_ACTIVE = 00000001
HIRT$M_BUSY = 00000002
HIRT$M_EMPTYPE = 00000010
HIRT$M_ERLOGIP = 00000080
HIRT$M_FE = 00000004
HIRT$M_MATCH = 00000008
HIRT$M_RCTFE = 00000100
HIRT$M_RCTFULL = 00000040
HIRT$M_RESCAN = 00000020
HIRT$V_ACTIVE = 00000000
HIRT$V_BUSY = 00000001
HIRT$V_EMPTYPE = 00000004
HIRT$V_ERLOGIP = 00000007
HIRT$V_FE = 00000002
HIRT$V_MATCH = 00000003
HIRT$V_RCTFE = 00000008
HIRT$V_RCTFULL = 00000006
HIRT$V_RESCAN = 00000005
HIRT$W_BOFF0 00000044 R 03
HIRT$W_BOFF1 00000046 R 03
HIRT$W_BOFF2 00000048 R 03
HIRT$W_BOFF3 0000004A R 03
HIRT$W_IOST 00000008 R 03
HIRT$W_IOWORST 0000000A R 03
HIRT$W_PAGENO 00000056 R 03
HIRT$W_PG0CNTNT 0000004C R 03
HIRT$W_PG1CNTNT 0000004E R 03
HIRT$W_PG2CNTNT 00000050 R 03
HIRT$W_PG3CNTNT 00000052 R 03
HIRT$W_SECTORNO 00000054 R 03
HIRT$W_STS 0000000E R 03
HIR_ERR_ONLINE 0000002D R 04
HIR_ERR_REPLACE 00000025 R 04
HIR_ERR_SEG1 00000034 R 04
HIR_ERR_SEG2 00000044 R 04
HIR_ERR_SEG3 0000004F R 04
HIR_ERR_TYPES 00000000 R 04
IOC$BROADCAST ******** X 02
IOC$CVT_DEVNAM ******** X 02
IRP$B_CARCON = 0000003C
IRP$B_EFN = 00000022
IRP$B_PRI = 00000023
IRP$B_RMOD = 0000000B
IRP$B_TYPE = 0000000A
IRP$K_LENGTH = 000000C4
IRP$L_ABCNT = 00000040
IRP$L_ARB = 00000058
IRP$L_AST = 00000010
IRP$L_ASTPRM = 00000014
IRP$L_BCNT = 00000032
IRP$L_CDT = 00000084
IRP$L_DIAGBUF = 0000004C
IRP$L_EXTEND = 00000054
IRP$L_FQFL = 00000060
IRP$L_IOQBL = 00000004
IRP$L_IOQFL = 00000000
IRP$L_IOSB = 00000024
IRP$L_IOST1 = 00000038
IRP$L_IOST2 = 0000003C
IRP$L_JNL_SEQNO = 00000048
IRP$L_MEDIA = 00000038
IRP$L_OBCNT = 00000044
IRP$L_PID = 0000000C
IRP$L_SEGVBN = 00000048
IRP$L_SEQNUM = 00000050
IRP$L_SVAPTE = 0000002C
IRP$L_TT_TERM = 0000003C
IRP$L_UCB = 0000001C
IRP$L_WIND = 00000018
IRP$Q_NT_PRVMSK = 00000040
IRP$W_ABCNT = 00000040
IRP$W_BCNT = 00000032
IRP$W_BOFF = 00000030
IRP$W_CHAN = 00000028
IRP$W_FUNC = 00000020
IRP$W_OBCNT = 00000044
IRP$W_SIZE = 00000008
```

```
IRP$W_STS                       =  0000002A
MAP_PAGE                           00000AAB  R      02
MMG$GL_SPTBASE                     ********    X    02
MSCP$B_BUFFER                   =  00000010
MSCP$B_FLAGS                    =  00000009
MSCP$B_OPCODE                   =  00000008
MSCP$K_OP_READ                  =  00000021
MSCP$K_OP_REPLC                 =  00000014
MSCP$K_OP_WRITE                 =  00000022
MSCP$K_ST_DATA                  =  00000008
MSCP$K_ST_SUCC                  =  00000000
MSCP$K_ST_WRTPR                 =  00000006
MSCP$L_BYTE_CNT                 =  0000000C
MSCP$L_CMD_REF                  =  00000000
MSCP$L_FRST_BAD                 =  0000001C
MSCP$L_LBN                      =  0000001C
MSCP$L_RBN                      =  0000000C
MSCP$M_MD_COMP                  =  00004000
MSCP$M_MD_ERROR                 =  00001000
MSCP$M_MD_EXPRS                 =  00008000
MSCP$M_MD_PRIMR                 =  00000001
MSCP$M_MD_SECOR                 =  00000200
MSCP$M_MD_SEREC                 =  00000100
MSCP$M_ST_MASK                  =  0000001F
MSCP$S_ST_MASK                  =  00000005
MSCP$V_EF_BBLKR                 =  00000007
MSCP$V_EF_ERLOG                 =  00000005
MSCP$V_ST_MASK                  =  00000000
MSCP$W_MODIFIER                 =  0000000A
MSCP$W_STATUS                   =  0000000A
NEXT                               00000B2D  R      02
OPA$UCB0                           ********    X    02
PDT$L_ALLOCMSG                  =  00000014
PDT$L_DEALLOMSG                 =  00000020
PDT$L_MAPIRP                    =  00000034
PDT$L_RCHMSGBUF                 =  00000044
PDT$L_UNMAP                     =  00000064
RCT$L_BAD_RBN                   =  00000014
RCT$L_LBN                       =  0000000C
RCT$L_RBN                       =  00000010
RCT$M_ALLOCATED                 =  20000000
RCT$M_BR                        =  00002000
RCT$M_FE                        =  00000080
RCT$M_NONPRIME                  =  10000000
RCT$M_RP1                       =  00008000
RCT$M_RP2                       =  00004000
RCT$M_UNUSABLE                  =  40000000
RCT$S_LBN                       =  0000001C
RCT$V_ALLOCATED                 =  0000001D
RCT$V_BR                        =  0000000D
RCT$V_FE                        =  00000007
RCT$V_LBN                       =  00000000
RCT$V_NULL                      =  0000001F
RCT$V_RP1                       =  0000000F
RCT$V_RP2                       =  0000000E
RCT$W_FLAGS                     =  00000008
RD$L_CDRP                       =  00000000
READ_RCT_BLOCK                     000009C3  R      02
REPLACE_CONNECT_FAILURE            0000080B  R      02
SCS$DEALL_RSPID                    ********    X    02
SCS$FIND_RDTE                      ********    X    02
SCS$UNSTALLUCB                     ********    X    02
SEARCH_FAIL                        00000B72  R      02
SEARCH_RCT                         00000ADB  R      02
SEARCH_RTN                         00000B7B  R      02
SIZ...                          =  00000001
SIZE                            =  00000045
SS$_BADRCT                      =  0000216C
SS$_CANCEL                      =  00000830
SS$_NORMAL                      =  00000001
SS$_WRITLCK                     =  0000025C
STEP10                             00000526  R      02
STEP11                             00000562  R      02
STEP12                             00000609  R      02
STEP13                             000006B0  R      02
STEP14                             000006D4  R      02
STEP15_A                           000006EF  R      02
STEP15_B                           00000717  R      02
STEP16                             00000783  R      02
STEP17                             000007C5  R      02
STEP18                             000007E6  R      02
STEP5                              00000399  R      02
STEP6                              000003B2  R      02
STEP7                              000003FB  R      02
STEP8                              000004B4  R      02
STEP9                              000004FE  R      02
STEPSIZ                         =  00000002
TEST_PATTERN                    =  B6DBCB6D
TEST_RCT_ENTRY                     00000B85  R      02
TYPSIZ                          =  00000007
UCB$B_DU_RBNPTRK                   ********    X    02
UCB$B_DU_RCTCPYS                   ********    X    02
UCB$L_CDDB                      =  000000BC
UCB$L_CDT                       =  000000C8
UCB$L_DU_USIZE                     ********    X    02
UCB$V_MSCP_WRTP                 =  0000000D
UCB$W_DEVSTS                    =  00000068
UCB$W_DU_LBNPTRK                   ********    X    02
UCB$W_DU_RCTSIZE                   ********    X    02
UCB$W_RWAITCNT                  =  00000056
VA$S_VPN                        =  00000015
VA$V_VPN                        =  00000009
WRITE_RCT_BLOCK                    00000909  R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000001F8 (504.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00000C9E (3230.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$300_HIRT	00000088 (136.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$301_HIR_ERRORS	00000056 (86.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	37	00:00:00.07	00:00:00.33
Command processing	139	00:00:00.44	00:00:02.55
Pass 1	724	00:00:22.06	00:01:13.75
Symbol table sort	0	00:00:03.17	00:00:09.39
Pass 2	407	00:00:05.60	00:00:41.25
Symbol table output	1	00:00:00.20	00:00:00.47
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1310	00:00:31.58	00:02:07.79

The working set limit was 2700 pages.
187469 bytes (367 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2953 non-local and 117 local symbols.
2652 source lines were read in Pass 1, producing 29 object records in Pass 2.
61 pages of virtual memory were used to define 59 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]DUTULIB.MLB;1	8
_255DUA28:[SYS.OBJ]LIB.MLB;1	31
_255DUA28:[SYSLIB]STARLET.MLB;2	9
TOTALS (all libraries)	48

3143 GETS were required to define 48 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:DUHIRT/OBJ=OBJ$:DUHIRT MSRC$:DUHIRT/UPDATE=(ENH$:DUHIRT)+EXECML$/LIB+LIB$:DUTULIB/LIB
```

DUHIRT
LIS

DUDRIVER
LIS